REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 2006	USARIEM Technical Report

4. TITLE AND SUBTITLE
Software Tool for Analysis of Variance of DNA Microarray Data

5. FUNDING NUMBERS

6. AUTHOR(S)
P. Khatri, D. Chen, J. Reifman, C.M. Lilly, L.A. Sonna

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Thermal and Mountain Medicine Division
U.S. Army Research Institute of Environmental Medicine
Natick, MA 01760-5007

8. PERFORMING ORGANIZATION REPORT NUMBER
T07-05

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Same as #7 above

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*
This report describes a software tool for two-way analysis of variance (ANOVA) with repeated measures on one factor for use in a multitude of problems, including the analysis of Affymetrix GeneChipTM microarray data. The proposed software has been used to analyze more than 22,000 probe sequences in less than 1 minute. The tool is entirely written in Java and, as a result, is platform-independent. The current implementation of the tool only allows for one-way and two-way ANOVA. However, the internal data structure is designed to be able to hold data for multi-way ANOVA. The output is written to a tab-delimited text file and, accordingly, the tool can be easily extended to save the results directly into a relational database.

14. SUBJECT TERMS
DNA Microarray Data, analysis of variance, gene sequence

15. NUMBER OF PAGES
88

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	Unclassified

USARIEM TECHNICAL REPORT T07-05

SOFTWARE TOOL FOR ANALYSIS OF VARIANCE OF DNA MICROARRAY DATA

Purvesh Khatri[1,2]
Dechang Chen[3]
Jaques Reifman[1]
Craig M. Lilly[4]
Larry A. Sonna[5,6]

[1]Bioinformatics Cell, Telemedicine and Advanced Technology Research Center,
U.S. Army Medical Research and Materiel Command, Fort Detrick, MD 21702

[2]Department of Computer Science
Wayne State University, Detroit, MI 48202

[3]Division of Epidemiology and Biostatistics
Uniformed Services University of the Health Sciences, Bethesda, MD 20814

[4]Division of Pulmonary and Critical Care Medicine
University of Massachusetts, Worcester, MA 01655

[5]Thermal and Mountain Medicine Division
U.S. Army Research Institute of Environmental Medicine, Natick, MA 01760

[6]Division of Pulmonary and Critical Care Medicine
University of Maryland School of Medicine, Baltimore, MD 21201

December 2006

U.S. Army Research Institute of Environmental Medicine
Natick, MA 01760-5007

TABLE OF CONTENTS

LIST OF TABLES

EXECUTIVE SUMMARY

This report describes a software tool for two-way analysis of variance (ANOVA) with repeated measures on one factor for use in a multitude of problems, including the analysis of Affymetrix GeneChip™ microarray data. The proposed software has been used to analyze more than 22,000 probe sequences in less than 1 minute.

The tool is entirely written in Java and, as a result, is platform-independent. The current implementation of the tool only allows for one-way and two-way ANOVA. However, the internal data structure is designed to be able to hold data for multi-way ANOVA. The output is written to a tab-delimited text file and, accordingly, the tool can be easily extended to save the results directly into a relational database.

INTRODUCTION

The development of increasingly powerful DNA microarrays for the study of gene expression has created a need for powerful computational tools with which to analyze the data. For example, a late-generation Affymetrix oligonucleotide array (the U133A) contains probe sets for 22,283 sequences. Each of these is, in theory, capable of detecting a change in expression for the sequence it detects independently of all other probe sets on the array. This means that any statistical test designed to detect individual sequence changes in expression will need to be repeated 22,283 times over, and the output will need to be formatted in a manner that is easily imported into software packages capable of handling large amounts of data, such as relational databases.

When simple experimental designs are involved, such as (for example) five to ten paired sets of samples, the mathematical analyses are fairly straightforward and can be performed readily with a commercially available spreadsheet, such as Microsoft Excel, which can perform a T-test and output the resulting P-value into the same row as the sequence under from which it was generated. For more complex research designs, however, such as comparisons between three or more means or two-factorial designs (e.g., multiple experimental conditions tested at varying time points), the use of spreadsheets becomes increasingly impractical due both to computational constraints (memory, processing power) and the difficulty of managing the output. For example, in Microsoft Excel, a one-way analysis of variance (ANOVA) generates a table of output, which greatly complicates the process of maintaining a one-to-one relationship between the input data and the output of the calculation.

A need exists for computational tools that can rapidly perform ANOVA on DNA microarray data. ANOVA is an excellent method for analyzing microarray data when multiple groups or factors are involved (such as treatment vs. time), as it permits statistical evaluation of changes in gene expression and variability in the array data (3;4). ANOVA has been productively used to calculate the contribution and significance of different treatments to changes in gene expression in published literature (e.g., see reference (1)).

There are several software tools that can perform ANOVA, including commercial programs, such as SAS (6) and SPSS (8), as well as free tools, such as the R programming language (11). However, the vast amount of data generated by even a comparatively small microarray experiment can unmask software limitations that are not always apparent when the number of variables under consideration is small. For example, SPSS generates large amounts of output other than P values for each ANOVA calculation; this feature is very useful when the number of independent calculations is small, but cumbersome to deal with when attempting to perform the same computation on hundreds to thousands of separate variables. Furthermore, the Windows Graphical User Interface versions of SPSS available when this project was started were not capable of performing thousands of two-way ANOVA calculations at one sitting on the hardware platforms available at the time. Conversely, console /

character-based statistical programming languages, such as R and SAS, are highly capable of performing the required analyses but have their own unique disadvantages. For example, the cost of a license to run a general-purpose program, such as SAS, makes it an unattractive option for a highly specialized requirement. The R programming language is available as freeware, but also suffers the performance limitations inherent to a general-purpose, interpreted language, which greatly limits the speed at which repetitive calculations can be performed. SPSS also has a built-in scripting language, but unlike R, which is available as freeware, a script written in SPSS is of value only to users who purchase a license to run SPSS.

We therefore decided to develop and implement a software tool that allows the user to perform one-way and two-way ANOVA with or without repeated measures for any number of instances. We chose to write the program in Java to maximize the number of platforms on which it can execute. The tool generates results as tab-delimited files that can be readily imported into relational databases.

This software tool will be maintained by the Bioinformatics Cell of the U.S. Army Medical Research and Materiel Command, Fort Detrick, Maryland.

SAMPLE DATA SET

Although the software allows the user to carry out two-way repeated measures ANOVA for any data set, it was written with a particular experiment in mind. The experimental design has three conditions (heat, cold, control) in which gene expression signals are measured at time points $T = 0$ and $T = 3$. The experiment is repeated six times, and the data from each trial are numbered from 1 to 6. There are a total of 22,283 gene sequences in each data set, whose signals are measured in each experiment. Each gene sequence is labeled in the database with a unique identifier (ID). An abstract representation of this data set for a single gene is shown in Table 1 (next page).

Table 1: Abstract representation of a sample data for a single gene, where Y_{ijk} is the measured expression of a gene in sample *i* for condition *j* at time *k*. Each gene is measured in 6 samples for 3 conditions at 2 time points.

Condition	Sample	Time	
		$T = 0$	$T = 3$
Heat	1	Y_{111}	Y_{112}
	2	Y_{211}	Y_{212}
	3	Y_{311}	Y_{312}
	4	Y_{411}	Y_{412}
	5	Y_{511}	Y_{512}
	6	Y_{611}	Y_{612}
Cold	1	Y_{121}	Y_{122}
	2	Y_{221}	Y_{222}
	3	Y_{321}	Y_{322}
	4	Y_{421}	Y_{422}
	5	Y_{521}	Y_{522}
	6	Y_{621}	Y_{632}
Control	1	Y_{131}	Y_{132}
	2	Y_{231}	Y_{232}
	3	Y_{331}	Y_{332}
	4	Y_{431}	Y_{432}
	5	Y_{531}	Y_{532}
	6	Y_{631}	Y_{632}

MATERIALS AND METHODS

TWO-WAY ANOVA WITH REPEATED MEASURES ON ONE FACTOR

The goal of our data analysis is to perform two-way, repeated measures ANOVA for each gene sequence. In our sample dataset, the between-subjects factor is *condition,* and the within-subjects repeated measures factor is *time*. For each gene sequence in the dataset, the following statistics are calculated:

- P-value for the main effect (*condition*).
- P-value for the within-subjects factor (*time*).
- P-value for the interaction (*time * condition*).

ONE-WAY ANOVA

In addition to the two-way repeated measures ANOVA, an additional secondary analysis is also performed. Because one of the factors is repeated, it is possible to derive a slope of the factor. Each gene is measured at two different time points in three different conditions, which allows us to calculate a slope for each gene in each condition.

- Slope(heat) = [(signal(heat) at T=3) - (signal(heat) at T=0)] / [3 - 0]
- Slope(cold) = [(signal(cold) at T=3) - (signal(cold) at T=0)] / [3 - 0]
- Slope(control) = [(signal(control) at T=3) - (signal(control) at T=0)] / [3 - 0]

In other words, for each gene in each of the six experiments, three derived variables are computable: Slope(heat), Slope(cold), Slope(control). Using these slope values we can perform a one-way ANOVA with *condition* (heat, cold, control) as the between-groups factor for each gene in the dataset. We compute:

- P-value for the main effect (*condition*).
- Post-hoc Tukey's test for the comparisons:
  - Heat vs. Control
  - Heat vs. Cold
  - Cold vs. Control

The following sections describe in detail the one-way and the two-way, repeated measures ANOVA models used for computing these results.

TWO-WAY ANOVA MODEL WITH REPEATED MEASURES ON ONE FACTOR

Suppose two factors, *A* and *B,* are involved in a study and repeated measures are taken on *B*, where *A* has *a* levels, *B* has *b* levels, and *n* subjects are available for each level of *A*. Let Y_{ijk} denote the observation for the *i-th* subject when *A* is at level *j* and *B* at level *k*. Let α_j denote the main effect of *A* at level *j*, β_k the main effect of *B* at level *k*, and $(\alpha\beta)_{jk}$ the interaction effect, when *A* is at level *j* and *B* at level *k, where, j=1, 2, ..., a; and, k=1, 2, ..., b*. In addition, let $\rho_{i(j)}$ denote the subject's main effect.

Assuming that no interactions exist between treatments and subjects, we have the following model (5):

$$Y_{ijk} = \mu_{...} + \rho_{i(j)} + \alpha_j + \beta_k + (\alpha\beta)_{jk} + \varepsilon_{ijk} \quad (1)$$

where, i) $\mu_{...}$ is a constant; ii) $\rho_{i(j)}$ is *iid* $N(0,\sigma_p^2)$ †; iii) α_j, β_k, and $(\alpha\beta)_{jk}$ are constants, such that $\Sigma\alpha_j = 0, \Sigma\beta_k = 0$, $\Sigma_j(\alpha\beta)_{jk} = 0$ for all k, and $\Sigma_k(\alpha\beta)_{jk} = 0$ for all j; iv) the error terms ε_{ijk} are *iid* $N(0, \sigma^2)$; and v) $\rho_{i(j)}$ and ε_{ijk} are independent.

In our example dataset, for each gene, we need to perform the two-way repeated measures ANOVA, where the between-subjects factor is the *condition* (heat, cold or control) and the within-subjects repeated measures factor is *time*.

The effect of each factor and their interactions are calculated as (5):

$$MSA = \frac{bn\sum_j \left(\overline{Y}_{.j.} - \overline{Y}_{...}\right)^2}{a-1} \quad (2)$$

$$MSB = \frac{an\sum_k \left(\overline{Y}_{..k} - \overline{Y}_{...}\right)^2}{b-1} \quad (3)$$

$$MSAB = \frac{n\sum_j\sum_k \left(\overline{Y}_{.jk} - \overline{Y}_{.j.} - \overline{Y}_{..k} + \overline{Y}_{...}\right)^2}{(a-1)(b-1)} \quad (4)$$

$$MSS(A) = \frac{b\sum_i\sum_j \left(\overline{Y}_{ij.} - \overline{Y}_{.j.}\right)^2}{a(n-1)} \quad (5)$$

$$MSB.S(A) = \frac{\sum_i\sum_j\sum_k \left(Y_{ijk} - \overline{Y}_{.jk} - \overline{Y}_{ij.} - \overline{Y}_{.j.}\right)^2}{a(n-1)(b-1)} \quad (6)$$

$$\overline{Y}_{...} = \frac{\sum_i\sum_j\sum_k Y_{ijk}}{nab} \quad (7)$$

† "*iid*" = independently and identically distributed

$$\overline{Y}_{.j.} = \frac{\sum_i \sum_k Y_{ijk}}{nb} \tag{8}$$

$$\overline{Y}_{..k} = \frac{\sum_i \sum_j Y_{ijk}}{na} \tag{9}$$

$$\overline{Y}_{.jk} = \frac{\sum_i Y_{ijk}}{n} \tag{10}$$

$$\overline{Y}_{ij.} = \frac{\sum_k Y_{ijk}}{b} \tag{11}$$

where $i = 1, 2, \ldots, n$; $j = 1, 2, \ldots, a$; and $k = 1, 2, \ldots, b$; n is the number of times the experiment is performed, a is the number of different values the factor for the main effect can have, and b is the number of different values for the repeated factor. In the case of our example data set, $n = 6$, $a = 3$, $b = 2$, and Y_{ijk} is the expression value of the gene in *i-th* sample in *j-th* condition at *k-th* time point.

The p-value for the main effect (i.e., the factor *condition* in our example) is calculated as the probability of $F \geq F^*$, where F is a random variable with numerator degrees of freedom equal to $a - 1$, denominator degrees of freedom equal to $a(n - 1)$, and

$$F^* = \frac{MSA}{MSS(A)}. \tag{12}$$

The p-value for the within-subjects factor (i.e., the factor *time* in our example) is calculated as the probability of $F \geq F^*$, where F is a random variable with numerator degrees of freedom equal to $b - 1$, denominator degrees of freedom equal to $a(n - 1)(b - 1)$, and

$$F^* = \frac{MSB}{MSB.S(A)}. \tag{13}$$

The p-value for the interaction is calculated as the probability of $F \geq F^*$, where F is a random variable with numerator degrees of freedom equal to $(a - 1)(b - 1)$, denominator degrees of freedom equal to $a(n - 1)(b - 1)$, and

$$F^* = \frac{MSAB}{MSB.S(A)}. \quad (14)$$

ONE-WAY ANOVA MODEL

Let us suppose there is only one factor involved in a study, where the factor has *r* levels, and n_i denotes the number of cases for the *i-th* factor level. The single-factor ANOVA model can be stated as follows (5):

$$Y_{ij} = \mu_i + \varepsilon_{ij} \quad (15)$$

where (i) Y_{ij} is the observation in the *j-th* trial for the *i-th* factor level (for $i = 1, 2, \ldots, r$ and $j = 1, 2, \ldots, n_i$); (ii) μ_i represents the mean response for the *i-th* factor level; and (iii) ε_{ij} are *iid* $N(0, \sigma^2)$.

As discussed before, in the example dataset, for each gene in each condition, a signal slopes can be computed and used for the one-way ANOVA computation with the condition as the between-groups factor. The signal slope for each gene in a given condition and a given sample can be calculated as:

$$Y_{ij} = \frac{T_{3ij} - T_{0ij}}{3-0}, \quad (16)$$

where T_{3ij} is the signal in condition *i* in sample *j* at time = 3, and T_{0ij} is the signal in condition *i* in sample *j* at time = 0.

After calculating the slopes, the data for each gene in Table 1 can be represented as shown in Table 2.

Table 2: Representation of the data for a gene after calculating the signal slopes for the given condition and the given sample, where Y_{ij} is the slope for condition *i* and sample *j*. For the example data set, we can calculate six slopes for each of the three conditions.

Condition	Slopes					
Heat (μ_1)	Y_{11}	Y_{12}	Y_{13}	Y_{14}	Y_{15}	Y_{16}
Cold (μ_2)	Y_{21}	Y_{22}	Y_{23}	Y_{24}	Y_{25}	Y_{26}
Control (μ_3)	Y_{31}	Y_{32}	Y_{33}	Y_{34}	Y_{35}	Y_{36}

Now we can calculate the p-values for the main effect (i.e., the *condition* in our example) using Equations *17* to *22* (5), defined as follows:

$$MSTR = \frac{\sum n_i \left(\overline{Y}_{i.} - \overline{Y}_{..}\right)^2}{r-1} \quad (17)$$

$$MSE = \frac{\sum\sum\left(Y_{ij} - \overline{Y}_{i.}\right)^2}{n_T - r} \tag{18}$$

$$\overline{Y}_{i.} = \frac{\sum_{j=1}^{n_i} Y_{ij}}{n_i} \tag{19}$$

$$\overline{Y}_{..} = \frac{\sum_i \sum_j Y_{ij}}{n_T} \tag{20}$$

$$n_T = \sum_i n_i \tag{21}$$

where $i = 1, 2, \ldots, r$; $j = 1, 2, \ldots, n_i$; r is the number of different values the factor for the main effect can have, and n_i is the maximum number of slopes for each different value of the factor for the main effect. For our example data set, $r = 3$ and $n_1 = n_2 = n_3 = 6$. Equation *19* calculates the average slope for each gene in each condition, and Equation *20* calculates the overall average slope for each gene.

Using the F-distribution, the p-value for the main effect (i.e., the factor *condition* in our example) is calculated as the probability of $F \geq F^*$, where F is a random variable with numerator degrees of freedom equal to $(r - 1)$, denominator degrees of freedom equal to $n_T - r$, and

$$F^* = \frac{MSTR}{MSE}. \tag{22}$$

TUKEY'S HSD COMPARISONS

The second goal of the one-way ANOVA is to do Post-hoc Tukey's HSD (Honestly Significant Difference) comparisons for all pairs. However, instead of calculating the p-value for each pair, the program provides confidence limits with family confidence coefficient of at least $1 - \alpha$. The confidence limits provide estimation of the range over which the difference of the mean value for the pair varies with the family confidence coefficient of at least $1 - \alpha$. If the confidence limits contain zero, the means of the two conditions are the same at some point. However, if the limits does not include zero, the means are never the same with the p-value of α. For our example data, consider that for a given gene, the mean signal value in condition heat is μ_1 and in condition cold is μ_2. Assume that the confidence limits (calculated as $\mu_1 - \mu_2$) are *[-100, 100]* and $\alpha = 0.05$. Since, the limits include zero, we can say that the means μ_1 and μ_2 are not different with *95%* confidence. However, if the confidence limits were either *[-100, -50]* or *[100, 1000]*, we can say that the means μ_1 and μ_2 are different with *95%* confidence (p-value of *0.05*).

The confidence limits for all pair-wise comparisons D = μ_1 - μ_2 with family confidence coefficient of at least $1 - \alpha$ are $\hat{D} \pm T \cdot S\{\hat{D}\}$, where

$$\hat{D} = \overline{Y}_i - \overline{Y}_j \tag{23}$$

$$S\{\hat{D}\} = \sqrt{MSE\left(\frac{1}{n_i} + \frac{1}{n_j}\right)} \tag{24}$$

$$T = \frac{1}{\sqrt{2}} q(1-\alpha; r, n_T - r) \tag{25}$$

In the case of our example dataset, $r = 3$, $n_T = 18$, and α of 0.01, 0.05, and 0.1, the function $q(1 - \alpha; r, n_T - r)$ in Equation *25* evaluates to 4.83, 3.67, and 3.14, respectively.

PROGRAM DESCRIPTION

The program is written in Java. As such, it is platform independent and runs on Windows, Linux, Mac OS, and Unix. To use the tool, the following files are required: The ANOVA Java program file itself, compiled as an executable Java program; a configuration file (described below, with an example shown in Appendix A) that supplies the program with the basic information it needs to perform the computation and allows the user to specify various parameters and the format of the input file; the data file to be analyzed; and a mathematical library publicly available by the Jakarta project (10), which is used to calculate the *F* distribution.

For users experienced in compiling Java programs, the source code is listed in Appendix B to this report. Executable code with sample input and configuration files can also be obtained directly from the authors (E-mail: jaques.reifman@us.army.mil, larry_sonna@yahoo.com or purvesh@cs.wayne.edu). A minimally-annotated configuration file, test data, and output files that can be used to verify program execution are included in Appendix C.

How to Use the Tool

In order to use the tool, Java must be installed on the user's computer. Java can be downloaded from Sun Microsystems (9). The following command can be used at the command-line to run the tool:

java anova.client.AnovaAnalysis1 <configuration file name> - d <# of digits>

The switch "-d" is optional and is followed by the number of digits that the user wishes the program to compute for each P value. The default is 8-digit precision.

Note that the Jakarta commons-math library should be in the "classpath" (the directory) so that Java can find it during execution. The classpath can be specified at

execution time by "-cp" switch. For instance, if commons-math library and the ANOVA code is in the same directory, the following command can be used:

java –cp "./commons-math-1.1.jar;./" anoava.client.AnovaAnalysis1
<configuration file name> (for Windows)

or

java –cp "./commons-math-1.1.jar:./" anova.client.AnovaAnalysis1
<configuration_file_name> (for Unix/Linux)

To start a command-line program on a Microsoft windows platform, click on "Start," and then "Run." Type "cmd" in the dialog box that pops up and press the "OK" button. A DOS window will open; the user will then need to change the directory (using the "cd" command) to the directory where the ANOVA tool is located. Once in that directory, the commands listed above can be entered to start the program. For the convenience of inexperienced Windows users, a batch program that executes the ANOVA tool is included in the electronic distribution (available from the authors).

On a Unix or Linux workstation, right-clicking the mouse and selecting "terminal" or "console" from the pop-up menu usually opens a command-line terminal.

The configuration file should be prepared as described in the following section. The results are saved in files as specified in the configuration file. The formats of the input file and the output file are described below.

Configuration File

Use of the ANOVA Java tool requires that the user supply the program with a configuration file. This file gives the program the basic information it needs to properly perform the desired computation. A sample file is included in Appendix A, and the required contents of the file are listed in Tables 3 and 4 (below). The ANOVA Java tool requires that the configuration file be a simple ASCII file; thus, if the configuration file is created using Microsoft Word or Notepad, it must be saved as a plain "text" file.

The format of the configuration file is similar to that of windows ".ini" files. The configuration file is divided into various sections, called "paragraphs." Each paragraph starts with the name of the paragraph enclosed between brackets, "[" and "]". This approach allows for easy addition of the new paragraphs in the future while keeping the new configuration file compatible with its previous versions.

Any line starting with "#" in the configuration file is treated as a comment line, i.e., does not contain instructions that the program will use.

The configuration parameters are divided into two parts: (1) global configuration parameters and (2) factor configuration parameters. The global configuration parameters are specified in the "config" paragraph, and each ANOVA factor is described in a "Factor*N*" paragraph, where *N* is an integer. Note that a "Factor"

paragraph refers to each index used to describe a data item, not the factors (in a statistical sense) used in the analysis. For instance, in the case of the sample data set in Table 1, each data item is described by three indices: (1) a sample number (1-to-6), (2) a condition (heat, cold, or control), and (3) a time point (t=0 or t=3). Of these three indices, only condition and time indices refer to the two ANOVA factors used in the analysis. However, in order for the program to know which data to pair up in time, the sample number factor is required. The integer number used to describe each "Factor" paragraph must start at 1 and should be incremented by 1 for each subsequent index. For instance, in the case of the sample data set, there are three indices used to describe a data item. Therefore, the configuration file must contain three "Factor" paragraphs, i.e., Factor1, Factore2, and Factor3.

Each paragraph (either "config" or "FactorN") assigns value to one or more parameters. The syntax to assign a value to a parameter is "*name = value.*" The parameters for a paragraph can be specified in any order. The paragraph name and the parameter name are case-insensitive.

The "config" paragraph contains the parameters described in Table 3, and the "Factor" paragraph parameters are described in Table 4. Appendix A provides a sample configuration file.

Table 3. Description of the parameters for the "config" paragraph.

Parameter Name	Description
NumOfFactors	The total number of indices used to describe each data item. It must be an integer number. For instance, in the case of our sample data set, NumOfFactors = 3.
DataFile	The location of the input file containing the data. The location can be either an absolute path or a relative path.
TwoWayOutputFile	File name in which the results of the two-way ANOVA should be stored.
HeadersForOneWayAnova	Factor name that should be used for column headers in the file specified by OneWayOutputFIle (see below).
OneWayOutputFile	File name in which the results of the one-way ANOVA should be stored.

Table 4: Description of parameters for the “factor” paragraph.

Parameter Name	Description
Name	Name of the index described by the paragraph. It can be any free-form string.
Column	The column number in the input file in which the value for the index can be found. The first column in the input file has index 1. The input file is the file specified by DataFile parameter in the "config" paragraph (See Table 3).
MaxNumOfIndex	The maximum value possible for the index.
Enumerate	Optional. If an index has categorical values, each category can be assigned an integer, starting at 1 and incremented by 1. When enumerating an index, the category name and its index must be separated by a tab character. See Appendix A for an example.

RESULTS

INPUT FORMAT

The input file is specified via the DataFile parameter of the "config" paragraph (see Table 3). The data for ANOVA is always a matrix, where each element of the matrix is a number. Therefore, the format of the input file for any ANOVA software requires that the input file describes one matrix element per line. For an example of the input file format, see Appendix A.

OUTPUT FORMAT

The ANOVA tool carries out one-way and two-way ANOVA computations on the data in the input file. The names of the output files for both of the analyses are specified by OneWayOutputFile and TwoWayOutputFile parameters of the "config" paragraph (see Table 3). The formats of the one-way ANOVA output file and the two-way ANOVA output file are described in Appendix A.

DESCRIPTION OF APPENDIX B

Appendix B provides a listing of the Java program along with its related routines, a list of the configuration file for the sample data set discussed in the report, and an input file for 20 genes with its associated one-way and two-way ANOVA output files. Appendix C contains a test dataset and the output resulting from application of the ANOVA tool to this dataset.

COMPARISON OF OUTPUT GENERATED BY THE JAVA PROGRAM TO THE OUTPUT GENERATED BY SPSS 10.0.5

To validate the Java software routine described in this report, we generated a test file of 90 sequences from our sample dataset and computed the two-way, repeated-measures ANOVA P values for condition, time, and the interactive factor condition*time using SPSS 10.0.5. The output was rounded to 15 significant digits (16 digits total, including the leading zero). The sample file used to validate the output against SPSS contained different data from the file used to code and debug the software tool (Appendix B). To insure that the computational algorithm would hold true not only for mid-range P values but also for extremes, we included genes in this validation test file that had P values, as measured by the Java program, both very close to zero, as well as close to 1. To verify that results would be comparable across hardware platforms, the computation with the Java program was performed on a different computer than the one on which the SPSS computation was performed.

The results are shown in Table 5. The differences between the two computations were negligibly small and within the margin of error expected from floating-point computation error (2). The comparison showed that the Java program output is

essentially the same as that obtained with one of the major commercial software statistical packages.

Table 5. Comparison of results generated by the JAVA ANOVA program to the results generated by SPSS 10.0.5 No more than four significant digits are reported in this table.

Table 5A. Raw data

	JAVA PROGRAM P VALUES			SPSS 10.0.5 P VALUES			DIFFERENCE		
Probe ID	COND	TIME	TIME * COND	COND	TIME	TIME * COND	COND	TIME	TIME * COND
200878_at	0.0362	0.0859	0.6661	0.0362	0.0859	0.6661	4.305E-11	-1.530E-11	-6.542E-12
201060_x_at	0.2106	0.3122	0.0329	0.2106	0.3122	0.0329	-6.273E-12	-1.475E-11	6.751E-11
201118_at	0.3434	0.0619	0.1243	0.3434	0.0619	0.1243	5.129E-12	-1.426E-11	-3.784E-11
201162_at	0.6202	0.5742	0.0001	0.6202	0.5742	0.0001	-2.817E-11	2.968E-12	1.683E-12
201163_s_at	0.4764	0.4796	0.6164	0.4764	0.4796	0.6164	-9.416E-11	9.589E-11	-3.160E-11
201206_s_at	0.7017	0.6065	0.6438	0.7017	0.6065	0.6438	-1.904E-12	8.835E-13	-1.352E-11
201858_s_at	0.3759	0.3434	0.4293	0.3759	0.3434	0.4293	2.426E-11	-4.325E-12	4.860E-12
201909_at	0.2241	0.7017	0.8814	0.2241	0.7017	0.8814	2.836E-10	1.355E-12	3.899E-12
202018_s_at	0.4326	0.6594	0.9795	0.4326	0.6594	0.9795	4.400E-12	7.755E-12	-1.266E-14
202611_s_at	0.1436	0.7317	0.0020	0.1436	0.7317	0.0020	-1.394E-11	2.370E-13	3.665E-12
203153_at	0.5695	0.0174	0.2065	0.5695	0.0174	0.2065	-6.065E-12	1.328E-11	-7.408E-12
203290_at	0.6791	0.1096	0.5627	0.6791	0.1096	0.5627	-4.216E-12	2.089E-10	-7.493E-12
203828_s_at	0.1447	0.2542	0.0073	0.1447	0.2542	0.0073	-1.320E-11	-1.450E-10	7.303E-11
203979_at	0.0826	0.0462	0.1427	0.0826	0.0462	0.1427	3.028E-12	-8.394E-11	-1.456E-11
204174_at	0.6158	0.9531	0.3105	0.6158	0.9531	0.3105	-3.216E-11	2.551E-13	1.500E-11
204351_at	0.6527	0.2599	0.9837	0.6527	0.2599	0.9837	-1.016E-11	-1.156E-10	-2.442E-15
204439_at	0.6672	0.1829	0.4941	0.6672	0.1829	0.4941	-6.307E-12	4.619E-11	-5.690E-11
204475_at	0.9726	0.5315	0.3444	0.9726	0.5315	0.3444	-8.116E-14	1.542E-11	4.962E-12
204669_s_at	0.9832	0.7536	0.8736	0.9832	0.7536	0.8736	-3.886E-15	9.881E-14	6.278E-12
204750_s_at	0.0173	0.7846	0.1477	0.0173	0.7846	0.1477	2.240E-12	-7.905E-12	-1.135E-11
204939_s_at	0.2446	0.1787	0.5278	0.2446	0.1787	0.5278	1.382E-10	5.678E-11	-2.143E-11
204970_s_at	0.5543	0.0079	0.6932	0.5543	0.0079	0.6932	-9.683E-12	3.952E-12	-2.577E-12
205000_at	0.1022	0.1543	0.0019	0.1022	0.1543	0.0019	-1.331E-10	1.442E-11	4.391E-12
205033_s_at	0.7948	0.2621	0.1250	0.7948	0.2621	0.1250	6.251E-12	-1.061E-10	-3.653E-11
205118_at	0.9538	0.7522	0.7113	0.9538	0.7522	0.7113	-1.966E-12	8.749E-14	-1.334E-12
205254_x_at	0.2553	0.5994	0.5408	0.2553	0.5994	0.5408	9.550E-11	1.118E-12	-1.456E-11
205403_at	0.3065	0.0158	0.3602	0.3065	0.0158	0.3602	1.713E-11	2.152E-11	3.885E-11
205557_at	0.4041	0.4963	0.0513	0.4041	0.4963	0.0513	1.041E-11	5.390E-11	7.260E-12
205576_at	0.3173	0.0357	0.0098	0.3173	0.0357	0.0098	1.201E-11	2.171E-12	2.371E-11
206385_s_at	0.1824	0.2003	0.6906	0.1824	0.2003	0.6906	-1.990E-11	2.018E-11	-2.827E-12
206522_at	0.5928	0.0976	0.4570	0.5928	0.0976	0.4570	-2.908E-12	-6.144E-12	-1.619E-10

206582_s_at	0.7456	0.3568	0.0095	0.7456	0.3568	0.0095	4.432E-11	-2.570E-12	2.668E-11
206676_at	0.2883	0.7038	0.8468	0.2883	0.7038	0.8468	3.130E-11	1.128E-12	4.777E-13
206697_s_at	0.0257	0.3305	0.5646	0.0257	0.3305	0.5646	1.824E-12	-7.179E-12	-7.059E-12
206700_s_at	0.0000	0.0311	0.0225	0.0000	0.0311	0.0225	-8.912E-12	4.630E-12	3.372E-12
206785_s_at	0.9406	0.0336	0.0010	0.9406	0.0336	0.0010	2.520E-14	3.014E-12	1.086E-11
206871_at	0.3615	0.0539	0.3966	0.3615	0.0539	0.3966	3.737E-11	-3.370E-11	1.302E-11
206877_at	0.0494	0.2246	0.0004	0.0494	0.2246	0.0004	8.887E-12	6.588E-12	6.510E-12
207177_at	0.6558	0.3032	0.6550	0.6558	0.3032	0.6550	-9.170E-12	-2.096E-11	-9.434E-12
207269_at	0.7231	0.8016	0.7690	0.7231	0.8016	0.7690	-8.561E-13	-3.171E-12	1.824E-11
207384_at	0.6949	0.1093	0.6136	0.6949	0.1093	0.6136	-2.429E-12	2.130E-10	-3.435E-11
207521_s_at	0.3442	0.7907	0.7873	0.3442	0.7907	0.7873	4.994E-12	-5.646E-12	8.638E-12
207802_at	0.3932	0.6210	0.4274	0.3932	0.6210	0.4274	1.444E-11	4.499E-13	5.144E-12
207824_s_at	0.6045	0.2386	0.9744	0.6045	0.2386	0.9744	-1.985E-12	-2.721E-10	-5.351E-14
207892_at	0.3600	0.9718	0.5086	0.3600	0.9718	0.5086	3.917E-11	-3.553E-14	-3.753E-11
207935_s_at	0.1461	0.4933	0.6721	0.1461	0.4933	0.6721	-1.226E-11	5.980E-11	-5.356E-12
208450_at	0.5385	0.1307	0.0387	0.5385	0.1307	0.0387	-1.556E-11	5.641E-11	3.111E-11
208470_s_at	0.0064	0.2684	0.5642	0.0064	0.2684	0.5642	3.905E-12	-8.262E-11	-7.150E-12
208602_x_at	0.4534	0.2518	0.6643	0.4534	0.2518	0.6643	-1.788E-10	-1.598E-10	-6.938E-12
208791_at	0.4954	0.0331	0.5268	0.4954	0.0331	0.5268	-5.485E-11	3.291E-12	-2.207E-11
209156_s_at	0.0112	0.0123	0.0528	0.0112	0.0123	0.0528	1.410E-11	6.701E-11	6.228E-12
209395_at	0.0415	0.7655	0.9683	0.0415	0.7655	0.9683	2.193E-11	3.941E-14	-1.980E-13
209602_s_at	0.9892	0.2026	0.4324	0.9892	0.2026	0.4324	0.000E+00	1.813E-11	4.417E-12
209840_s_at	0.1848	0.1449	0.1737	0.1848	0.1449	0.1737	-1.800E-11	2.461E-11	-2.883E-11
210116_at	0.1871	0.1538	0.4936	0.1871	0.1538	0.4936	-1.634E-11	1.482E-11	-5.774E-11
210321_at	0.5831	0.0343	0.7447	0.5831	0.0343	0.7447	-3.957E-12	2.695E-12	4.569E-11
210362_x_at	0.5903	0.3790	0.3951	0.5903	0.3790	0.3951	-3.152E-12	-3.114E-11	1.365E-11
210690_at	0.6915	0.0524	0.8771	0.6915	0.0524	0.8771	-2.740E-12	-4.007E-11	5.075E-12
210875_s_at	0.2178	0.5598	0.0404	0.2178	0.5598	0.0404	3.556E-10	5.361E-12	2.491E-11
211010_s_at	0.3193	0.8268	0.5607	0.3193	0.8268	0.5607	1.124E-11	-6.907E-13	-7.970E-12
211372_s_at	0.5618	0.0342	0.8184	0.5618	0.0342	0.8184	-7.687E-12	2.765E-12	2.117E-12
211560_s_at	0.7413	0.5930	0.8448	0.7413	0.5930	0.8448	5.167E-11	1.461E-12	5.340E-13
211583_x_at	0.2750	0.6497	0.5231	0.2750	0.6497	0.5231	4.892E-11	1.147E-11	-2.458E-11
212268_at	0.9348	0.0056	0.7017	0.9348	0.0056	0.7017	4.818E-14	3.245E-12	-1.899E-12
212736_at	0.0288	0.2236	0.5193	0.0288	0.2236	0.5193	1.245E-10	6.876E-12	-2.751E-11
212750_at	0.7852	0.2238	0.9559	0.7852	0.2238	0.9559	9.442E-12	6.809E-12	-1.490E-12
212768_s_at	0.5540	0.2664	0.1439	0.5540	0.2664	0.1439	-9.761E-12	-8.930E-11	-1.374E-11
213069_at	0.0034	0.6334	0.2542	0.0034	0.6334	0.2542	1.071E-11	2.340E-13	9.899E-11
213172_at	0.6264	0.4470	0.6480	0.6264	0.4470	0.6480	-2.328E-11	-2.598E-12	-1.183E-11
213906_at	0.0081	0.3101	0.0845	0.0081	0.3101	0.0845	4.887E-11	-1.603E-11	2.651E-12
214219_x_at	0.6005	0.7319	0.2025	0.6005	0.7319	0.2025	-2.264E-12	2.833E-13	-8.709E-12
214567_s_at	0.2668	0.4752	0.7056	0.2668	0.4752	0.7056	6.447E-11	1.114E-10	-1.651E-12
215047_at	0.2101	0.0830	0.2754	0.2101	0.0830	0.2754	-6.411E-12	-1.922E-11	4.824E-11
216858_x_at	0.8672	0.3129	0.1632	0.8672	0.3129	0.1632	1.384E-13	-1.432E-11	-5.370E-12
217022_s_at	0.1488	0.7041	0.1598	0.1488	0.7041	0.1598	-1.077E-11	1.116E-12	-6.309E-12
217147_s_at	0.5464	0.2553	0.5463	0.5464	0.2553	0.5463	-1.230E-11	-1.390E-10	-1.233E-11
217374_x_at	0.7659	0.3472	0.7137	0.7659	0.3472	0.7137	2.060E-11	-3.729E-12	-1.221E-12
217636_at	0.8302	0.8488	0.2350	0.8302	0.8488	0.2350	1.172E-12	-1.490E-13	1.929E-10

219003_s_at	0.1396	0.3449	0.1838	0.1396	0.3449	0.1838	-1.702E-11	-4.083E-12	-1.881E-11
219311_at	0.4003	0.7822	0.2177	0.4003	0.7822	0.2177	1.166E-11	-8.987E-12	3.575E-10
219529_at	0.0006	0.1329	0.2117	0.0006	0.1329	0.2117	1.365E-12	4.946E-11	-6.016E-12
219911_s_at	0.0046	0.9174	0.3660	0.0046	0.9174	0.3660	3.382E-12	-6.561E-14	3.267E-11
219922_s_at	0.6813	0.8565	0.2853	0.6813	0.8565	0.2853	-3.912E-12	-7.661E-15	3.464E-11
219948_x_at	0.3751	0.5325	0.8177	0.3751	0.5325	0.8177	2.483E-11	1.484E-11	2.190E-12
219978_s_at	0.0000	0.5662	0.0158	0.0000	0.5662	0.0158	1.278E-12	4.178E-12	3.377E-12
220330_s_at	0.9907	0.5827	0.6795	0.9907	0.5827	0.6795	0.000E+00	2.195E-12	-4.165E-12
220418_at	0.7624	0.6845	0.4561	0.7624	0.6845	0.4561	2.361E-11	2.687E-12	-1.658E-10
220785_at	0.4850	0.1540	0.1907	0.4850	0.1540	0.1907	-7.385E-11	1.468E-11	-1.408E-11
221586_s_at	0.0726	0.2021	0.1936	0.0726	0.2021	0.1936	6.538E-12	1.857E-11	-1.248E-11
222299_x_at	0.0269	0.8243	0.6629	0.0269	0.8243	0.6629	1.695E-10	-8.288E-13	-7.283E-12

Table 5B. Summary of Data

		P values			
Program	**Factor**	Lowest	Highest	Mean	Standard deviation
JAVA	COND	2.60E-06	0.9907	0.4222	0.2973
	TIME	5.58E-03	0.9718	0.3873	0.2839
	TIME*COND	8.82E-05	0.9837	0.4353	0.2991
SPSS	COND	2.60E-06	0.9907	0.4222	0.2973
	TIME	5.58E-03	0.9718	0.3873	0.2839
	TIME*COND	8.82E-05	0.9837	0.4353	0.2991
Difference	COND	-1.79E-10	3.56E-10	1.08E-11	6.40E-11
	TIME	-2.72E-10	2.13E-10	-1.90E-12	6.08E-11
	TIME*COND	-1.66E-10	3.57E-10	2.85E-12	5.50E-11

COMPARISON OF THE OUTPUT GENERATED BY THE JAVA PROGRAM TO OUTPUT GENERATED BY THE "R" PROGRAMMING LANGUAGE 2.1.1

To further validate the output of the ANOVA Java program, we compared the output it generated on a full dataset (22,283 sequences) to that generated by a routine written in the R programming language (v 2.1.1). As before, the computations were run on different computers, with 15 significant digits of output generated (16 digits total). As shown in Table 6, the differences between the outputs of the two programs were again negligibly small and attributable to floating point errors.

Importantly, the processing time required by the two programs running on the same computer was markedly different. On a late-model computer running a 2.8 GHz Pentium D dual-core processor under Windows XP, the ANOVA Java program required less than 60 seconds to compute the results of the dataset (which contained a total of 802,188 data points). By contrast, the R routine, which was designed only to compute

the P values for 2-way, RM ANOVA (and not 1-way ANOVA on time slopes or Tukey's HSD), required about 6 ¾ hours to execute.

On rare occasions, the ANOVA Java program generated P values of less than zero. Of 22,283 sequences analyzed in the test dataset, six had P values for COND of less than zero, the smallest value being -3.09×10^{-7}. In the R programming language, the corresponding P values for these six sequences were all positive but smaller than 5×10^{-9}. P values this small are, for practical purposes, highly statistically significant and no different from zero.

Table 6. Summary of a comparison between the outputs of the JAVA ANOVA program to the results generated by R 2.1.1

		P values			
Program	**Factor**	Lowest	Highest	Mean	Standard deviation
JAVA	COND	-3.08530E-07	0.99992	0.37032	0.31364
	TIME	1.97614E-08	1.00000	0.41854	0.30518
	COND*TIME	2.64314E-09	0.99995	0.42541	0.30594
R	COND	3.54267E-12	0.99992	0.37032	0.31364
	TIME	1.96174E-08	1.00000	0.41854	0.30518
	COND*TIME	4.44393E-09	0.99995	0.42541	0.30594
Difference	COND	-3.08668E-07	7.07311E-06	3.74096E-10	4.82288E-08
	TIME	-3.31890E-10	2.81597E-10	6.27451E-13	5.43576E-11
	COND*TIME	-1.80080E-09	3.23315E-10	-4.66212E-13	6.80459E-11

DISCUSSION

The ANOVA Java tool described herein was developed because of limitations inherent to popular general statistical packages that made them unattractive choices for performing two-way ANOVA on Affymetrix microarray data. These limitations included cost (SAS), speed of execution (R programming language), and user labor (SPSS). By contrast, the ANOVA Java tool allows the user to rapidly analyze any number of instances, limited only by the amount of physical and virtual memory available on the user's computer. Furthermore, the ANOVA Java tool is very fast – it was able to perform a computation on a dataset at a speed that was two orders of magnitude faster than a routine written in R. The output files generated by the tool are tab-delimited files that can readily be imported into almost any major relational database management system, such as Oracle, SQL server, Microsoft Access, etc., thus facilitating output management. Finally, the tool gives the user the option to determine the precision to which each P value is computed, though for practical purposes, most will find five to six digits sufficient.

At present, the ANOVA Java tool only allows one-way ANOVA on slopes and two-way repeated measures ANOVA. However, the data structure it uses is very generic and should be readily capable of being expanded to allow performance of multi-factorial ANOVA. Furthermore, since the ANOVA tool is entirely implemented in Java, it can take advantage of the platform-independent database connection mechanism provided by Java through JDBC (7). Using JDBC the tool can be expanded to read input data from the database and store the results directly to the database.

An additional limitation to the ANOVA Java tool is that it does not perform assumption checking on the data. This is more of a theoretical than practical limitation at present. ANOVA methods in general (and paired, repeated-measures ANOVA methods in particular) are fairly robust with regard to deviations from underlying assumptions methods. Additionally, when the numbers of replicates are low, the ability of existing statistical methods to reliably detect deviations from normality or homogeneity of variance is often quite low.

Another important limitation to the tool is that it was designed with a specific experiment in mind, and some features of the program (such as the time interval denominator used to compute slopes in the one-way ANOVA routine) can only be modified by direct modification of the source code.

In summary, we have designed and implemented a platform-independent software tool that can rapidly and accurately compute the results of multifactorial, repeated-measures microarray experiments. The program accepts text input and generates output that can easily be imported into popular commercial databases, with a degree of precision that is user-defined. The tool thus overcomes several important limitations that are evident when one attempts to use general-purpose software packages to perform ANOVA on microarray data.

REFERENCES

1. Adamson, C., N. Maitra, J. Bahl, K. Greer, S. Klewer, J. Hoying, and E. Morkin. Regulation of gene expression in cardiomyocytes by thyroid hormone and thyroid hormone analogs 3,5-diiodothyropropionic acid and CGS 23425 [N-[3,5-dimethyl-4-(4'-hydroxy-3'-isopropylphenoxy)-phenyl]-oxamic acid]. *J. Pharmacol.Exp.Ther.* 311: 164-171, 2004.

2. Goldberg, D. What every computer scientist should know about floating-point arithmetic. *ACM Computing Surveys* 23: 5-48, 1991.

3. Kerr, M. K. and G. A. Churchill. Experimental design for gene expression microarrays. *Biostatistics.* 2: 183-201, 2001.

4. Kerr, M. K., M. Martin, and G. A. Churchill. Analysis of variance for gene expression microarray data. *J Comput.Biol.* 7: 819-837, 2000.

5. Neter, J., M. Kutner, C. Nachtschiem, and W. Wasserman. Applied Linear Statistical Models. New York, NY, WCB / McGraw-Hill. 1996.

6. SAS Institute, Inc. http://www.sas.com/

7. Speegle GD. JDBC: Practical guide for Java programmers. San Francisco, CA, Morgan Kaufmann Publishers. 2002.

8. SPSS, Inc. http://www.spss.com/

9. Sun Mircrosystems, Inc. http://java.sun.com

10. The Jakarta Project. Commons-math: the Jakarata mathematics library. http://www.jakarta.apache.org/commons/math

11. The R Project for Statistical Computing. http://www.r-project.org/

APPENDIX A

ILLUSTRATIVE EXAMPLE OF CONFIGURATION FILE FORMAT

```
[config]
# Number of factors to consider in ANOVA.
NumOfFactors = 3
# Character that separates the columns in the input file.
# The value of this parameter must be enclosed in " and ".
ColumnSeparator = ","
# Input file.
DataFile = input.txt
TwoWayOutputFile = SampleTwoWayAnova.txt
OneWayOutputFile = SampleOneWayAnova.txt
HeadersForOneWayAnova = Factor2

# Now describe each factor individually.
# A factor paragraph has the format FactorN, where N is substituted by an
# integer.
# The number of Factor paragraphs must be equal to the value of NumOfFactors
# parameter in the config paragraph (see above).

# Factor "sample" has 6 experiments, and it is specified in the second column of
# the input file.

[Factor1]
Name = sample
Columns = 2
MaxNumOfIndex = 6
# Example of a factor that has categorical values.
# The factor is enumerated by assigning an integer to each category.
# The integers must start at 1 and increment by 1.
# Note that the category name and the corresponding integer are separated by a
# tab character.
# For example, "heat" and "1" are separated by a tab.
[Factor2]
Name = condition
Columns = 3
MaxNumOfIndex = 3
Enumerate = heat	1	cold	2	control 3

[Factor3]
Name = time
Columns = 4
MaxNumOfIndex = 2
# Note that time here is treated as a categorical parameter, with two values,
# 1 and 2, corresponding to T=0 and T=3, respectively.  The time interval denominator
# used to compute slopes is presently not user-modifiable
```

ILLUSTRATIVE EXAMPLE OF INPUT FILE FORMAT

For the sample dataset, the experiment is performed 6 times for 3 conditions. In each trial, each gene is measured at two different time points in every condition. Therefore, we will have 36 values for each gene. The matrix for one gene is described in Table 1. However, in the matrix each row provides measured intensities of the gene at two different time points for a given sample and a given condition. The matrix in Table 1 is essentially a 3-dimensional matrix of size 3 x 6 x 2. It is a representation of a 3-dimensional matrix in 2-dimension space. Note that the matrix in Table 1 contains 3 embedded smaller tables, one for each condition. The problem of representing the data in matrix form becomes difficult as we increase the number of factors (i.e., the dimensions of the matrix).

However, the matrix in Table 1 can be represented in a different format with *N + 1* columns, where the first *N* columns contain indices for the number of experiment, the condition, and the time point, respectively, and the last column contains the expression value of the gene. This format can describe one gene only. If we add the unique ID of each gene as the first column, we can now represent the data for any number of genes. The new table now contains *N + 2* columns. Table 7 shows the format with N + 2 columns separated by commas, where the first column is the unique ID of the gene, the second column is the number of the experiment, the third column is the condition, the fourth column is the time (treated as a categorical variable with 1 corresponding to the first time point and 2 corresponding to the second time point), and the last column is the expression value of the gene. Note that this format requires that each instance in the input file be assigned a unique label even if the input file contains only one instance.

Table 7. Sample input file.

```
AFFX-BioB-5_at,1,Control,1,470.90
AFFX-BioB-5_at,1,Control,2,455.50
AFFX-BioB-5_at,1,Heat,1,486.70
AFFX-BioB-5_at,1,Heat,2,464.40
AFFX-BioB-5_at,1,Cold,1,378.40
AFFX-BioB-5_at,1,Cold,2,380.00
AFFX-BioB-5_at,2,Control,1,370.40
AFFX-BioB-5_at,2,Control,2,399.30
AFFX-BioB-5_at,2,Heat,1,418.20
AFFX-BioB-5_at,2,Heat,2,454.70
AFFX-BioB-5_at,2,Cold,1,395.30
AFFX-BioB-5_at,2,Cold,2,425.40
AFFX-BioB-5_at,3,Control,1,393.40
AFFX-BioB-5_at,3,Control,2,392.70
AFFX-BioB-5_at,3,Heat,1,386.60
AFFX-BioB-5_at,3,Heat,2,428.60
AFFX-BioB-5_at,3,Cold,1,282.90
AFFX-BioB-5_at,3,Cold,2,331.80
AFFX-BioB-5_at,4,Control,1,347.80
```

AFFX-BioB-5_at,4,Control,2,389.80
AFFX-BioB-5_at,4,Heat,1,393.80
AFFX-BioB-5_at,4,Heat,2,443.60
AFFX-BioB-5_at,4,Cold,1,354.20
AFFX-BioB-5_at,4,Cold,2,432.30
AFFX-BioB-5_at,5,Control,1,339.50
AFFX-BioB-5_at,5,Control,2,373.10
AFFX-BioB-5_at,5,Heat,1,402.40
AFFX-BioB-5_at,5,Heat,2,524.00
AFFX-BioB-5_at,5,Cold,1,346.10
AFFX-BioB-5_at,5,Cold,2,360.60
AFFX-BioB-5_at,6,Control,1,354.50
AFFX-BioB-5_at,6,Control,2,314.40
AFFX-BioB-5_at,6,Heat,1,376.80
AFFX-BioB-5_at,6,Heat,2,1097.10
AFFX-BioB-5_at,6,Cold,1,484.50
AFFX-BioB-5_at,6,Cold,2,419.20

ILLUSTRATIVE EXAMPLE OF OUTPUT FILE FORMAT FOR TWO-WAY ANOVA WITH REPEATED MEASURES ON ONE FACTOR

An example of the output generated by the two-way ANOVA algorithm is listed in Table 8. The program generates p-values for main effect (condition), the repeated measures factor (time), and the interaction of condition x time.

Table 8. Sample output file for two-way ANOVA with repeated measures on one factor.

Gene Name	p-value for condition	p-value for time	p-value for interaction
AFFX-BioC-5_at	0.242995	0.301010	0.525759
AFFX-Crex-3_at	0.135440	0.083110	0.429870
AFFX-PheX-M_at	0.571269	0.881806	0.122794
AFFX-DapX-M_at	0.940233	0.689611	0.081855
AFFX-ThrX-M_at	0.704193	0.595223	0.769351
AFFX-LysX-5_at	0.844022	0.777465	0.868368

ILLUSTRATIVE EXAMPLE OF OUTPUT FILE FORMAT FOR ONE-WAY ANOVA

The sample output for one-way ANOVA is shown in Table 9. The tool provides confidence intervals for three different p-values: 0.01, 0.05, and 0.1. The order in which the columns are presented is as follows:

Column 1 Sequence name
Column 2 P value for condition
Subsequent columns: Tukey HSD confidence intervals for subgroup comparisons at P-values 0.01, 0.05, and 0.1. The number of columns will depend upon the number of subgroups being compared, as define by "Factor2" in the configuration file. In our sample computation, there are three subgroups (heat, cold, control) and hence, three possible comparisons (heat vs. cold, heat vs. control, cold vs. control). So, in our output example:

Columns 3, 4, 5: Confidence intervals, Index (1) vs. Index (2) (in this case, Heat vs. Cold) for the following P values:
Column 3: P=0.01
Column 4: P=0.05
Column 5: P=0.10
Columns 6, 7, 8: Confidence intervals, Index (1) vs. Index (3) (in this case, Heat vs. Control)
Column 6: P=0.01
Column 7: P=0.05
Column 8: P=0.10
Columns 9, 10, 11: Confidence intervals, Index (2) vs. Index (3) (in this case, Cold vs. Control) for the following P values:
Column 9: P=0.01
Column 10: P=0.05
Column 11: P=0.10

Table 9. Illustrative example of output file format for one-way ANOVA.

Name	p value for condition	Heat - Cold (p = 0.01)	Heat - Cold (p = 0.05)	Heat - Cold (p = 0.1)	Heat - Control (p = 0.01)	Heat - Control (p = 0.05)	.....
AFFX-BioC-5_at	0.5257599	[-203.58465649, 388.7957676]	[-132.44994718, 317.66105829]	[-99.94874379, 285.1598549]	[-216.10687871, 376.27354538]	[-144.9721694, 305.13883607]	.....
AFFX-CreX-3_at	0.4298707	[-2749.89818953, 5577.72041175]	[-1749.89429952, 4577.71652174]	[-1292.99597046, 4120.81819269]	[-2754.72596731, 5572.89263397]	[-1754.7220773, 4572.88874397]	.....
AFFX-DapX-M_at	0.0818554	[-9.65480618, 53.5770284]	[-2.06174944, 45.98397166]	[1.407492, 42.51473022]	[-25.1270284, 38.10480618]	[-17.53397166, 30.51174944]	.....
AFFX-LysX-5_at	0.868369	[-12.3651477, 14.87625881]	[-9.0939229, 11.60503402]	[-7.59931157, 10.11042268]	[-14.48181437, 12.75959215]	[-11.21058957, 9.48836735]	.....
AFFX-PheX-M_at	0.1227942	[-1.53957245, 6.45068356]	[-0.58008001, 5.49119112]	[-0.14169122, 5.05280233]	[-2.10068356, 5.88957245]	[-1.14119112, 4.93008001]	.....
AFFX-ThrX-M_at	0.7693516	[-17.9959178, 25.38480669]	[-12.78663825, 20.17552714]	[-10.40653639, 17.79542528]	[-17.41258446, 25.96814002]	[-12.20330492, 20.75886048]	.....

.....	Heat - Control (p = 0.1)	Cold - Control (p = 0.01)	Cold - Control (p = 0.05)	Cold - Control (p = 0.1)
.....	[-112.47096601, 272.63763267]	[-308.71243427, 283.66798982]	[-237.57772495, 212.53328051]	[-205.07652156, 180.03207712]
.....	[-1297.82374824, 4115.99041491]	[-4168.63707842, 4158.98152286]	[-3168.63318841, 3158.97763285]	[-2711.73485935, 2702.0793038]
.....	[-14.06473022, 27.042508]	[-47.08813951, 16.14369507]	[-39.49508277, 8.55063833]	[-36.02584133, 5.08139688]
.....	[-9.71597824, 7.99375602]	[-15.73736993, 11.50403659]	[-12.46614513, 8.23281179]	[-10.9715338, 6.73820046]
.....	[-0.70280233, 4.49169122]	[-4.55623911, 3.43401689]	[-3.59674667, 2.47452445]	[-3.15835789, 2.03613566]
.....	[-9.82320306, 18.37875861]	[-21.10702891, 22.27369558]	[-15.89774936, 17.06441603]	[-13.5176475, 14.68431417]

THIS PAGE LEFT BLANK INTENTIONALLY

APPENDIX B

PROGRAM LISTING

The java code to run an ANOVA analysis has been laid out in two packages, namely, "anova.client" and "anova.util". The package "anova.client" contains one file named "AnovaAnalysis1.java", which is the main application. The list of the files contained in the package "anova.util", is as follows:

1. DataFile.java
2. Factor.java
3. iniElement.java
4. IniFile.java
5. IniParagraph.java
6. InputFile.java
7. OutputFile.java

The application also uses an external library named "commons-math-1.0-RC1.jar", available from apache [7]. The actual code contained in the files listed above is as follows:

AnovaAnalysis1.java

```
/*
 * Created on Aug 28, 2003
 */
package anova.client;

/**
 * @author Purvesh Khatri
 */
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Date;
import java.util.Iterator;
import java.util.StringTokenizer;

import org.apache.commons.math.MathException;
import org.apache.commons.math.distribution.DistributionFactory;
import org.apache.commons.math.distribution.FDistribution;

import anova.util.DataFile;
import anova.util.Factor;
import anova.util.IniFile;

public class AnovaAnalysis1
```

```
{
  private static int numOfSignificantDigits = 8;

  public static void main(String [] args)
  {
    if (args.length < 1)
    {
      System.err.println("USAGE: java anova.client.AnovaAnalysis <config_file>"
          + " [-d <number of significant digits>]");
      System.err.println(
          "Also make sure commons-math library is in classpath.");
      System.err.println(
          "You may need to modify the classpath using -cp argument.");
      System.exit(0);
    }

    IniFile ini = null;

    // Read the configuration file.
    try
    {
      ini = new IniFile(args[0]);
    }
    catch (IOException e)
    {
      System.out.println(e.getMessage());
      System.exit(1);
    }

    try
    {
      if (args.length > 1 && args[1].equals("-d"))
      {
        AnovaAnalysis1.numOfSignificantDigits = Integer.parseInt(args[2], 10);
      }
    }
    catch (NumberFormatException nfe)
    {
      nfe.printStackTrace();
      System.err.println("NumberFormatException: Invalid number of "
          + "significant digits specified: " + args[2]);
      System.err.println("\tIt must be an integer number.");
      System.err.println("Using default number of significant digits.");
      AnovaAnalysis1.numOfSignificantDigits = 8;
    }
```

```
  // Create an object to parse the input file, based on configuration file.
  DataFile dataFile = new DataFile(ini);

  // Now actually parse the file.
  Date start = new Date();
  System.err.print("Reading file...");
  dataFile.readDataFile();

  Date readFile = new Date();
  System.err.println("Done. Time taken: "
     + ((readFile.getTime() - start.getTime()) / 1000) + " sec");

  if (  !ini.getProfile("config", "twowayoutputfile").equals(""))
    doTwoWayAnalysis(dataFile, ini, dataFile.getSizeArray());

  Date endTwoWay = new Date();

  if ( !ini.getProfile("config", "twowayoutputfile").equals(""))
    System.err.println("Two way ANOVA done. Time for analysis: "
       + ((endTwoWay.getTime() - readFile.getTime()) / 1000) + " sec");

  if (  !ini.getProfile("config", "onewayoutputfile").equals(""))
    doOneWayAnalysis(dataFile, ini, dataFile.getSizeArray(), true);

  Date endOneWay = new Date();

  if (  !ini.getProfile("config", "onewayoutputfile").equals(""))
    System.err.println("One way ANOVA done. Time for analysis: "
       + ((endOneWay.getTime() - endTwoWay.getTime()) / 1000) + " sec");

  System.err.println("              Total time: "
     + ((endOneWay.getTime() - start.getTime()) / 1000) + " sec");
}

/**
 * @param value
 *        Floating-point number that needs to be rounded off
 * @param places
 *        Number of significant digits to which value should be rounded off
 * @return Returns the rounded off number.
 */
public static double round(double value, int places)
{
  long factor = (long) Math.pow(10, places);

  // Shift the decimal the correct number of places
```

```
  // to the right.
  value = value * factor;

  // Round to the nearest integer.
  long tmp = Math.round(value);

  // Shift the decimal the correct number of places
  // back to the left.
  return (double) tmp / factor;
}

private static void printHeaders(PrintWriter pw, IniFile iniFile)
{
  String headerFactor = iniFile.getProfile("config", "HeadersForOneWayAnova");
  String headersLine = iniFile.getProfile(headerFactor, "enumerate");
  if (headersLine == null || headersLine.equals(""))
    throw new NullPointerException(headerFactor + " must define headers "
        + "using Enumerate");

  StringTokenizer stk = new StringTokenizer(headersLine.substring(
     headersLine.indexOf('=') + 1));

  // the headersLine must be tokenized in even number of tokens
  String [] headers = new String [stk.countTokens() / 2];
  for (int i = 0; i < headers.length; i++)
  {
    headers[i] = stk.nextToken().trim();
    stk.nextToken();
  }

  for (int i = 0; i < headers.length; i++)
  {
    for (int j = i + 1; j < headers.length; j++)
    {
      // The order of p-values is hard-codes because we will always calculate
      // confidence intervals in this order.
      // see doTukeyComparison().
      pw.print("\t" + headers[i] + " - " + headers[j] + " (p = 0.01)");
      pw.print("\t" + headers[i] + " - " + headers[j] + " (p = 0.05)");
      pw.print("\t" + headers[i] + " - " + headers[j] + " (p = 0.1)");
    }
  }
}

private static void doOneWayAnalysis(DataFile dataFile, IniFile iniFile,
   int [] sizeArray, boolean doTukeyComparisons)
```

```
{
 PrintWriter pw = null;
 try
 {
  pw = new PrintWriter(new FileWriter(
     iniFile.getProfile("config", "onewayoutputfile")));
  pw.print("Name\tp value for condition");

  if (doTukeyComparisons)
   printHeaders(pw, iniFile);

  pw.println();
 }
 catch (IOException ioe)
 {
  System.err.println(ioe);
  System.exit(4);
 }

 Iterator instanceIterator = dataFile.getKeySetIterator();

 int countConditionPValue = 0;

 while (instanceIterator.hasNext())
 {
  // Get the next instance.
  String name = (String) instanceIterator.next();
  Factor factor = dataFile.get(name);

  int [] indexArray = new int [3];

  int i = 0, j = 0;

  // calculate means for each conditions.
  double [] conditionMean = new double [sizeArray[1]]; // Yi.
  for (j = 0; j < sizeArray[1]; j++)
  {
   for (i = 0; i < sizeArray[0]; i++)
   {
    indexArray[0] = i;
    indexArray[1] = j;
    indexArray[2] = 0;
    double f1 = ((Double) factor.get(indexArray)).doubleValue();
    indexArray[2] = 1;
    double f2 = ((Double) factor.get(indexArray)).doubleValue();
    conditionMean[j] += ((f2 - f1) / (3));
```

```
  }
  conditionMean[j] = conditionMean[j] / (i);
}

// calculate global mean
double globalMean = 0.0; // Y..
for (j = 0; j < sizeArray[1]; j++)
{
  for (i = 0; i < sizeArray[0]; i++)
  {
    indexArray[0] = i;
    indexArray[1] = j;
    indexArray[2] = 0;
    double f1 = ((Double) factor.get(indexArray)).doubleValue();
    indexArray[2] = 1;
    double f2 = ((Double) factor.get(indexArray)).doubleValue();
    globalMean += ((f2 - f1) / (3));
  }
}
globalMean = globalMean / (i * j);

double mstr = 0.0;
for (j = 0; j < sizeArray[1]; j++)
{
  mstr += (Math.pow(conditionMean[j] - globalMean, 2) * sizeArray[0]);
}
mstr = mstr / (sizeArray[1] - 1);

double mse = 0.0;
for (j = 0; j < sizeArray[1]; j++) // For each condition
{
  for (i = 0; i < sizeArray[0]; i++) // each measurement in each condition
  {
    indexArray[0] = i;
    indexArray[1] = j;
    indexArray[2] = 0;
    double f1 = ((Double) factor.get(indexArray)).doubleValue();
    indexArray[2] = 1;
    double f2 = ((Double) factor.get(indexArray)).doubleValue();
    mse += Math.pow(((f2 - f1) / (3)) - conditionMean[j], 2);
  }
}
mse = mse / ((sizeArray[0] * sizeArray[1]) - sizeArray[1]);

try
{
```

```
      // Now calculate p-values for condition.
      DistributionFactory factory = DistributionFactory.newInstance();

      FDistribution fDist = factory.createFDistribution(sizeArray[1] - 1,
         (sizeArray[0] * sizeArray[1]) - sizeArray[1]);
      double fStar = mstr / mse;
      double conditionPValue = 1 - fDist.cumulativeProbability(fStar);
      if (conditionPValue <= 0.01)
        countConditionPValue++;

      pw.print(name + "\t");

      if (conditionPValue <= 1e-10)
        pw.print(0.0);
      else
        pw.print(round(conditionPValue,
           AnovaAnalysis1.numOfSignificantDigits));
    }
    catch (MathException me)
    {
      System.err.println("MathException for instance " + name);
      me.printStackTrace();
      System.err.println("Skipping the instance " + name + "...");
    }
    if (doTukeyComparisons)
    {
      doTukeyComparison(conditionMean, mse, sizeArray, pw);
    }

    pw.println();
  } // end of while loop.

  pw.close();
}

private static void doTukeyComparison(double [] conditionMean, double mse,
    int [] sizeArray, PrintWriter pw)
{
  double [] g = {4.83, 3.67, 3.14};

  int i, j, k;

  for (i = 0; i < conditionMean.length; i++)
  {
    for (j = i + 1; j < conditionMean.length; j++)
    {
```

```
      for (k = 0; k < g.length; k++)
      {
        double approxD = conditionMean[i] - conditionMean[j];
        double approxStdD = Math.sqrt(
            mse * (1 / (double) sizeArray[0] + 1 / (double) sizeArray[0]));
        double t = g[k] / Math.sqrt(2);

        double lowerBound = approxD - (t * approxStdD);
        double upperBound = approxD + (t * approxStdD);

        pw.print("\t["
            + round(lowerBound, AnovaAnalysis1.numOfSignificantDigits) + ", "
            + round(upperBound, AnovaAnalysis1.numOfSignificantDigits) + "]");
      }
    }
  }
}

private static void doTwoWayAnalysis(DataFile dataFile, IniFile iniFile,
    int [] sizeArray)
{
  PrintWriter pw = null;
  try
  {
    pw = new PrintWriter(new FileWriter(
        iniFile.getProfile("config", "twowayoutputfile")));
    pw.println("Probe\tp value for condition\tp value for time\t "
        + "p value for interaction");
  }
  catch (IOException ioe)
  {
    System.err.println(ioe);
    System.exit(4);
  }

  Iterator instanceIterator = dataFile.getKeySetIterator();

  int countConditionPValue = 0;
  int countTimePValue = 0;
  int countInteractionPValue = 0;

  while (instanceIterator.hasNext())
  {
    String name = (String) instanceIterator.next();
    Factor factor = dataFile.get(name);
```

```
int [] indexArray = new int [3];

int i = 0, j = 0, k = 0;

// calculate means for each conditions.
double [] conditionMean = new double [sizeArray[1]]; // Y.j.
for (j = 0; j < sizeArray[1]; j++)
{
 for (i = 0; i < sizeArray[0]; i++)
 {
  for (k = 0; k < sizeArray[2]; k++)
  {
   indexArray[0] = i;
   indexArray[1] = j;
   indexArray[2] = k;
   conditionMean[j] += ((Double) factor.get(indexArray)).doubleValue();
  }
 }
 conditionMean[j] = conditionMean[j] / (k * i);
}

// calculate global mean
double globalMean = 0.0; // Y...
for (j = 0; j < sizeArray[1]; j++)
{
 for (i = 0; i < sizeArray[0]; i++)
 {
  for (k = 0; k < sizeArray[2]; k++)
  {
   indexArray[0] = i;
   indexArray[1] = j;
   indexArray[2] = k;
   globalMean += ((Double) factor.get(indexArray)).doubleValue();
  }
 }
}
globalMean = globalMean / (i * j * k);

// calculate means for each time point.
double [] timeMean = new double [sizeArray[2]]; // Y..k
for (k = 0; k < sizeArray[2]; k++)
{
 for (i = 0; i < sizeArray[0]; i++)
 {
  for (j = 0; j < sizeArray[1]; j++)
  {
```

```
      indexArray[0] = i;
      indexArray[1] = j;
      indexArray[2] = k;
      timeMean[k] += ((Double) factor.get(indexArray)).doubleValue();
    }
  }
  timeMean[k] = timeMean[k] / (j * i);
}

// calculate contime time interaction mean.
// Y.jk
double [][] conditionTimeMean = new double[sizeArray[1]][sizeArray[2]];
for (k = 0; k < sizeArray[2]; k++)
{
  for (j = 0; j < sizeArray[1]; j++)
  {
    for (i = 0; i < sizeArray[0]; i++)
    {
      indexArray[0] = i;
      indexArray[1] = j;
      indexArray[2] = k;
      conditionTimeMean[j][k] +=
        ((Double) factor.get(indexArray)).doubleValue();
    }
    conditionTimeMean[j][k] = conditionTimeMean[j][k] / (i);
  }
}

// Yij.
double [][] sampleConditionMean = new double [sizeArray[0]] [sizeArray[1]];
for (i = 0; i < sizeArray[0]; i++)
{
  for (j = 0; j < sizeArray[1]; j++)
  {
    for (k = 0; k < sizeArray[2]; k++)
    {
      indexArray[0] = i;
      indexArray[1] = j;
      indexArray[2] = k;
      sampleConditionMean[i][j] +=
        ((Double) factor.get(indexArray)).doubleValue();
    }
    sampleConditionMean[i][j] = sampleConditionMean[i][j] / (k);
  }
}
```

```
// Now calculate mean sum of squares.
double msa = 0.0;
for (j = 0; j < sizeArray[1]; j++)
{
  msa += Math.pow(conditionMean[j] - globalMean, 2);
}
msa = (msa * sizeArray[2] * sizeArray[0]) / (sizeArray[1] - 1);

double msb = 0.0;
for (k = 0; k < sizeArray[2]; k++)
{
  msb += Math.pow(timeMean[k] - globalMean, 2);
}
msb = (msb * sizeArray[1] * sizeArray[0]) / (sizeArray[2] - 1);

double msab = 0.0;
for (j = 0; j < sizeArray[1]; j++)
{
  for (k = 0; k < sizeArray[2]; k++)
  {
    msab += Math.pow(conditionTimeMean[j][k] - conditionMean[j]
       - timeMean[k] + globalMean, 2);
  }
}
msab = (msab * sizeArray[0]) / ((sizeArray[1] - 1) * (sizeArray[2] - 1));

double mssa = 0.0;
for (i = 0; i < sizeArray[0]; i++)
{
  for (j = 0; j < sizeArray[1]; j++)
  {
    mssa += Math.pow(sampleConditionMean[i][j] - conditionMean[j], 2);
  }
}
mssa = (mssa * sizeArray[2]) / (sizeArray[1] * (sizeArray[0] - 1));

double msbsa = 0.0;
for (i = 0; i < sizeArray[0]; i++)
{
  for (j = 0; j < sizeArray[1]; j++)
  {
    for (k = 0; k < sizeArray[2]; k++)
    {
      indexArray[0] = i;
      indexArray[1] = j;
      indexArray[2] = k;
```

```
      msbsa += Math.pow(((Double) factor.get(indexArray)).doubleValue()
         - conditionTimeMean[j][k] - sampleConditionMean[i][j]
         + conditionMean[j], 2);
    }
   }
}
msbsa = msbsa / (sizeArray[1] * (sizeArray[0] - 1) * (sizeArray[2] - 1));

// Now calculate p-values for condition.
DistributionFactory factory = DistributionFactory.newInstance();

FDistribution fDist = factory.createFDistribution(sizeArray[1] - 1,
    sizeArray[1] * (sizeArray[0] - 1));
double fStar = msa / mssa;
double conditionPValue = -1.0;
try
{
  conditionPValue = 1 - fDist.cumulativeProbability(fStar);
  if (conditionPValue <= 1e-10)
    conditionPValue = 0.0;
  if (conditionPValue <= 0.01)
    countConditionPValue++;
}
catch (MathException me)
{
  System.err.println("MathException for instance " + name);
  me.printStackTrace();
  System.err.println("Setting condition p-value for instance " + name
      + " as -1.0 in the output file.");
  conditionPValue = -1;
}

fDist.setNumeratorDegreesOfFreedom(sizeArray[2] - 1);
fDist.setDenominatorDegreesOfFreedom(sizeArray[1] * (sizeArray[0] - 1)
    * (sizeArray[2] - 1));
fStar = msb / msbsa;
double timePValue = -1.0;
try
{
  timePValue = 1 - fDist.cumulativeProbability(fStar);
  if (timePValue <= 1e-10)
    timePValue = 0.0;
  if (timePValue <= 0.01)
    countTimePValue++;
}
```

```
      catch (MathException me)
      {
        System.err.println("MathException for instance " + name);
        me.printStackTrace();
        System.err.println("Setting time p-value for instance " + name
            + " as -1.0 in the output file.");
        timePValue = -1;
      }

      fDist.setNumeratorDegreesOfFreedom((sizeArray[1] - 1)
          * (sizeArray[2] - 1));
      fDist.setDenominatorDegreesOfFreedom(sizeArray[1] * (sizeArray[0] - 1)
          * (sizeArray[2] - 1));
      fStar = msab / msbsa;
      double interactionPValue = -1.0;
      try
      {
        interactionPValue = 1 - fDist.cumulativeProbability(fStar);
        if (interactionPValue <= 1e-10)
          interactionPValue = 0.0;
        if (interactionPValue <= 0.01)
          countInteractionPValue++;
      }
      catch (MathException me)
      {
        System.err.println("MathException for instance " + name);
        me.printStackTrace();
        System.err.println("Setting interaction p-value for instance " + name
            + " as -1.0 in the output file.");
        interactionPValue = -1;
      }

      pw.println(name + "\t"
          + round(conditionPValue, AnovaAnalysis1.numOfSignificantDigits)
          + "\t" + round(timePValue, AnovaAnalysis1.numOfSignificantDigits)
          + "\t"
          + round(interactionPValue, AnovaAnalysis1.numOfSignificantDigits));
    }

    pw.println("No. of Genes (P <= 0.01)\t" + countConditionPValue + "\t"
        + countTimePValue + "\t" + countInteractionPValue);
    pw.close();
  }
}
```

DataFile.java

```
/*
 * Created on Aug 28, 2003
 */
package anova.util;

/**
 * @author Purvesh Khatri
 */
import java.io.*;
import java.util.*;

public class DataFile
{
  private IniFile iniFile;

  private int [] sizeArray;

  private LinkedHashMap dataTable;

  public DataFile(IniFile ini)
  {
    iniFile = ini;
    dataTable = new LinkedHashMap();
  }

  /**
   * Reads the input file. This is the public interface provided by DataFile
   * class.
   */
  public void readDataFile()
  {
    int numOfFactors = 0;
    try
    {
      numOfFactors = Integer.parseInt(iniFile.getProfile("config",
          "numoffactors"));
    }
    catch (NumberFormatException nfe)
    {
      System.err.println(
          "DataFile: Invalid number of factors specified."
              + "\nNumOfFactors must be an integer number. "
              + "Please correct your configuration file.");
      System.err.println("Specified NumOfFactors: "
```

```
        + iniFile.getProfile("config", "numoffactors"));
    System.exit(1);
  }

  // find out how the columns in the file are separated.
  String columnSeparator = iniFile.getProfile("config", "columnseparator");
  columnSeparator =
    columnSeparator.substring(1, columnSeparator.length() - 1);

  // Find out the dimensions of the data matrix.
  sizeArray = createSizeArray(numOfFactors);

  BufferedReader br = null;

  try
  {
    br = new BufferedReader(new FileReader(
        iniFile.getProfile("config", "datafile")));
  }
  catch (IOException ioe)
  {
    System.err.println("DataFile: Error opening the datafile.");
    System.err.println("Please check file name: "
        + iniFile.getProfile("config", "datafile"));
  }

  readFile(br);
}

private int [] createSizeArray(int numOfFactors)
{
  List paragraphNames = new ArrayList(iniFile.getParagraphNames());

  Collections.sort(paragraphNames);

  int [] sizeArray = new int [numOfFactors];

  int j = 0;
  int maxNumOfIndex = 0;
  for (int i = 0; i < paragraphNames.size(); i++)
  {
    String name = (String) paragraphNames.get(i);
    if (name.toLowerCase().startsWith("factor"))
    {
      try
      {
```

```
        maxNumOfIndex = Integer.parseInt(
            iniFile.getProfile(name, "MaxNumOfIndex"));
        }
        catch (NumberFormatException nfe)
        {
          System.err.println(
            "DataFile: Invalid number of indices specified."
              + "\nMaxNumOfIndex must be an integer number. "
              + "Please correct your configuration file "
              + "for the following paragraph.");
          System.err.println("Erroneous Paragraph: " + name);
          System.exit(1);
        }

        sizeArray[j] = maxNumOfIndex;
        j++;
      }
    }

    return sizeArray;
  }

  private void readFile(BufferedReader br)
  {
    String s = null;

    try
    {
      while ((s = br.readLine()) != null)
      {
        processInstance(s);
      }
    }
    catch (IOException ioe)
    {
      System.err.println("DataFile: Error reading data file.");
      System.err.println("Error encountered when reading: " + s);
      System.exit(2);
    }
  }

  private void processInstance(String instanceStr)
  {
    StringTokenizer stk = new StringTokenizer(instanceStr, iniFile.getProfile(
        "config", "ColumnSeparator"));
```

```
  // first column is always instance id or name.
  String name = stk.nextToken();

  int columnNum = 2;
  int [] indexArray = new int [Integer.parseInt(
     iniFile.getProfile("config", "numoffactors"))];

  for (int i = 0; i < indexArray.length; i++)
  {
   indexArray[i] = interpretIndex(stk.nextToken(), columnNum) - 1;
   columnNum++;
  }

  Factor factor = (Factor) dataTable.get(name);

  if (factor == null)
  {
   factor = new Factor(sizeArray);
  }

  factor.add(indexArray, new Double(stk.nextToken().trim()));
  dataTable.put(name, factor);
}

private int interpretIndex(String indexStr, int columnNum)
{
  List paragraphNames = new ArrayList(iniFile.getParagraphNames());

  Collections.sort(paragraphNames);

  List factorList = new ArrayList(Integer.parseInt(
     iniFile.getProfile("config", "numoffactors")));
  for (int i = 0; i < paragraphNames.size(); i++)
  {
   String name = (String) paragraphNames.get(i);
   if (name.toLowerCase().startsWith("factor"))
   {
    factorList.add(name);
   }
  }

  for (int i = 0; i < factorList.size(); i++)
  {
   int column = -1;
   try
   {
```

```
      column = Integer.parseInt(iniFile.getProfile(
         (String) factorList.get(i), "Columns"));
      }
      catch (NumberFormatException nfe)
      {
       System.err.println("DataFile: Error parsing the column number for "
          + factorList.get(i));
       System.err.println("Erroneous line in config file: "
          + iniFile.getProfile((String) factorList.get(i), "Columns"));
      }

      if (column != columnNum)
       continue;
      else
      {
       if (iniFile.getProfile(
          (String) factorList.get(i), "Enumerate").equals(""))
       {
        try
        {
         int index = Integer.parseInt(indexStr);
         return index;
        }
        catch (NumberFormatException nfe)
        {
         System.err.println(
            "DataFile: Non-integer number specified in column "
            + columnNum);
         System.exit(3);
        }
       }
       else
        return interpretEnumeration(indexStr,
          iniFile.getParagraph((String) factorList.get(i)));
      }
     }

     return -1;
    }

    private int interpretEnumeration(String indexStr, iniParagraph paragraph)
    {
     String inputStr = indexStr.toLowerCase();
     String enumStr = paragraph.tagValue("Enumerate");

     StringTokenizer stk = new StringTokenizer(enumStr, " \t");
```

```
  int enumValue = -1;
  while (stk.hasMoreTokens())
  {
   String enumName = stk.nextToken().trim().toLowerCase();
   try
   {
    enumValue = Integer.parseInt(stk.nextToken().trim());
   }
   catch (NumberFormatException nfe)
   {
    System.err.println("DataFile: Error interpreting enumeration");
    System.err.println("Erroneous paragraph is: " + paragraph.getName());
    System.err.println("Erroneous line is: "
       + paragraph.tagValue("enumerate"));
    System.exit(2);
   }

   if (enumName.equals(inputStr))
    break;
  }

  return enumValue;
}

public void writeDataFile(String fileName)
{
  try
  {
   PrintWriter pw = new PrintWriter(new FileWriter(fileName));

   Iterator iterator = dataTable.keySet().iterator();

   while (iterator.hasNext())
   {
    String name = (String) iterator.next();
    Factor factor = (Factor) dataTable.get(name);

    int [] indexArray = new int [Integer.parseInt(
       iniFile.getProfile("config", "numoffactors"))];
    for (int i = 0; i < sizeArray[0]; i++)
    {
     indexArray[0] = i;
     for (int j = 0; j < sizeArray[1]; j++)
     {
      indexArray[1] = j;
```

```
            for (int k = 0; k < sizeArray[2]; k++)
            {
              indexArray[2] = k;
              pw.println(name + "," + (indexArray[0] + 1) + ","
                 + (indexArray[1] + 1) + "," + (indexArray[2] + 1) + ","
                 + factor.get(indexArray));
            }
          }
        }
      }

      pw.close();
    }
    catch (IOException ioe)
    {
      System.err.println(ioe);
    }
  }

  public Iterator getKeySetIterator()
  {
    return dataTable.keySet().iterator();
  }

  public Factor get(String name)
  {
    return (Factor) dataTable.get(name);
  }

  public int [] getSizeArray()
  {
    return sizeArray;
  }
}
```

Factor.java

```
/*
 * Created on Aug 28, 2003
 */
package anova.util;

/**
 * @author Purvesh Khatri
 */
import java.util.ArrayList;
import java.util.List;

public class Factor
{
  /*
   * hasChildren is only set in the constructor and
   * is never modified. It is strictly a flag to maintain the structure
   * of the class Factor.
   */
  private boolean hasChildren;

  private List childrenList;

  public Factor(int [] sizeArray)
  {
    childrenList = new ArrayList(sizeArray[0]);

    if (sizeArray.length > 1)
    {
      hasChildren = true;

      /*
       * create new size array for the children of this factor object.
       * The new size array removes the size element from the beginning
       * of the size array.
       */
      int [] newSizeArray = getNewArray(sizeArray);

      // create and add children of this Factor instance.
      for (int i = 0; i < sizeArray[0]; i++)
        childrenList.add(i, new Factor(newSizeArray));
    }
  }

  public Object get(int [] indexArray)
```

```
{
  // If it does not have children, return the object at given index.
  if ( !hasChildren)
    return childrenList.get(indexArray[0]);

  // Well, we have some children!
  // Get the child at the given index i.e. indexArray[0].
  Factor factor = (Factor) childrenList.get(indexArray[0]);

  // Now reduce the indexArray size by one, i.e.,
  // copy indices from 1 to (length - 1)
  // to a new array.
  int [] newIndexArray = getNewArray(indexArray);

  // Ask the child to retrieve the given object.
  return factor.get(newIndexArray);
}

private int [] getNewArray(int [] array)
{
  int [] newSizeArray = new int [array.length - 1];
  for (int i = 0; i < newSizeArray.length; i++)
    newSizeArray[i] = array[i + 1];

  return newSizeArray;
}

public void add(int [] indexArray, Double value)
{
  if ( !hasChildren)
  {
    childrenList.add(value);
    return;
  }

  Factor factor = (Factor) childrenList.get(indexArray[0]);

  int [] newIndexArray = getNewArray(indexArray);

  factor.add(newIndexArray, value);
}

public void getList(int [] array, List list)
{
  if (array[0] == -1)
  {
```

```
    if ( !hasChildren)
    {
      for (int i = 0; i < childrenList.size(); i++)
        list.add(childrenList.get(i));
      return;
    }

    array = getNewArray(array);

    for (int i = 0; i < childrenList.size(); i++)
    {
      Factor factor = (Factor) childrenList.get(i);
      factor.getList(array, list);
    }
    return;
  }

  Factor factor = (Factor) childrenList.get(array[0]);
  int [] newArray = getNewArray(array);

  factor.getList(newArray, list);
 }
}
```

IniFile.java

```
package anova.util;

import java.io.File;
import java.io.IOException;
import java.util.Enumeration;
import java.util.Hashtable;
import java.util.Set;

/**
 * this class handles Windows style ini-files of the form
 * [paragraphname]
 * key1=value1
 * key2=value2
 *
 * [another_paragraphname]
 * keyx = vals
 * tag=foo
 * etc. it will, however, work on any platform
 */

public class IniFile
{
  private String filename, path, fullpath;

  private Hashtable paragraphs;

  private iniParagraph para;

  private InputFile f;

  /**
   * Opens an existing iniFile. If it does not exist, an empty one is created and
   * opened
   *
   * @param -
   *          pathname
   * @param filename,
   *          usually xxx.ini
   */
  public IniFile(String pathname, String fname) throws IOException
  {
    filename = fname;
    path = pathname;
    if (path.length() == 0)
```

```
    fullpath = filename;
  else
    fullpath = path + File.separator + filename;
  checkFile(fullpath);
  f = new InputFile(fullpath);
  getParagraphs();
}

private void checkFile(String fname) throws IOException
{
  File fl = new File(fname);
  if (  !fl.exists())
  {
    OutputFile outf = new OutputFile(fname);
    outf.close();
  }
}

/**
 * Opens an ini file in the current directory
 *
 * @param -
 *          filename, usually xxx.ini
 */
public IniFile(String fname) throws IOException
{
  filename = fname;
  path = "";
  fullpath = filename;
  checkFile(fullpath);
  f = new InputFile(filename);
  getParagraphs();
}

private synchronized void getParagraphs()
{
  paragraphs = new Hashtable(10); // make room for object table
  if ( !f.checkErr())
  {
    para = new iniParagraph(f);

    if ( !para.error())
    {
      paragraphs.put(para.getName(), para);
      while ((para.nextLine() != null) && (para.nextLine().length() > 0))
      {
```

```
        para = new iniParagraph(f, para.nextLine());
        paragraphs.put(para.getName(), para);
      }
    }
    f.close();
  }
}

/**
 * Returns an iniParagraph instance for the given paraName.
 *
 * @param Paragraph
 *        name.
 * @return An iniParagraph reference, if paraName is found, else null.
 */
public iniParagraph getParagraph(String paraName)
{
  return (iniParagraph) paragraphs.get(paraName);
}

/**
 * Returns a set of all paragraph names.
 *
 * @return java.util.Set containing all paragraph sections.
 */
public Set getParagraphNames()
{
  return paragraphs.keySet();
}

private synchronized void putParagraphs() throws IOException
{
  OutputFile f = new OutputFile(fullpath);
  Enumeration xenum = paragraphs.elements();
  while (xenum.hasMoreElements())
  {
    iniParagraph p = (iniParagraph) xenum.nextElement();
    f.println("[" + p.getName() + "]");

    Enumeration e = p.getTags();
    while (e.hasMoreElements())
    {
      String tag = (String) e.nextElement();
      f.println(tag.toLowerCase() + "=" + p.tagValue(tag));
    }
  }
```

```
  f.close();
}

/**
 * Gets a profile string from an ini file
 *
 * @param para -
 *        paragraph name in any case
 * @param entry -
 *        name of entry in any case
 * @return String value of this entry
 */
public String getProfile(String para, String entry)
{
  iniParagraph ini = null;
  ini = (iniParagraph) paragraphs.get(para.toLowerCase());
  if (ini != null) // now look for entry in that paragraph
    return ini.tagValue(entry).trim();
  else
    return "";
}

/**
 * Gets a profile string from an ini file returns default value if no such
 * entry exists
 *
 * @param para -
 *        paragraph name in any case pattern
 * @param entry -
 *        name of entry in any case pattern
 * @param default -
 *        default value for parameter
 * @return String value of this entry
 */

public String getProfile(String para, String entry, String defalt)
{
  String value = getProfile(para, entry);
  if (value.length() < 1)
    value = defalt;
  return value.trim();
}

/**
 * Puts a profile entry into the ini file
 *
```

```
     * @param para
     *        -paragraph name
     * @param tag -
     *        name of entry
     * @param value -
     *        value of entry
     */
    public void putProfile(String para, String tag, String value)
        throws IOException
    {
      boolean found = false;
      iniParagraph p = null;
      Enumeration xenum = paragraphs.elements();
      while (xenum.hasMoreElements() && !found)
      {
        p = (iniParagraph) xenum.nextElement();
        found = (para.equalsIgnoreCase(p.getName()));
      }
      if (found)
      {
        p.setValue(tag, value);
      }
      else
      {
        p = new iniParagraph(para);
        paragraphs.put(p.getName(), p);
        p.setValue(tag.toLowerCase(), value);
      }
      putParagraphs(); // rewrite entire ini-file
    }
}
```

iniParagraph.java

```
package anova.util;

import java.util.ArrayList;
import java.util.Collections;
import java.util.Enumeration;
import java.util.Hashtable;
import java.util.List;

/** A private class used by IniFile */
public class iniParagraph
{
  public List getSortedTags()
  {
    Enumeration chipEnum = getTags();

    ArrayList list = new ArrayList();

    while (chipEnum.hasMoreElements())
    {
      list.add((String) chipEnum.nextElement());
    }

    Collections.sort(list);

    return list;
  }

  private String paragraph_name;

  private InputFile f;

  private Hashtable tags;

  private String line;

  private boolean error_flag;

  public iniParagraph(InputFile fl)
  {
    f = fl;
    String s = getNextLine();
    read_to_nextparagraph(s);
  }
```

```
public Enumeration getTags()
{
  return tags.keys();
}

public Hashtable getHash()
{
  return tags;
}

public void setValue(String tag, String value)
{
  tags.put(tag.toLowerCase(), value); // may replace old value
}

private void read_to_nextparagraph(String s)
{
  error_flag = false;
  tags = new Hashtable();

  // The while loop looks for [paragraph] line.
  while ((s != null) && ( !s.startsWith("[")))
    s = getNextLine();

  // remove brackets
  if (s != null)
  {
    s = s.substring(1); // all but first char
    int i = s.indexOf("]");
    if (i > 0)
    {
      s = s.substring(0, i);
      store_paragraph(s);
    }
    else
    {
      error_flag = true;
    }
  }
  else
  {
    error_flag = false;
  }
}

public String getName()
```

```
{
  return paragraph_name.toLowerCase();
}

public String tagValue(String tagname)
{
  String ans;

  ans = (String) tags.get(tagname.toLowerCase());
  if (ans == null)
    return "";
  else
    return ans;
}

private void store_paragraph(String s)
{
  String next;
  iniElement ini;
  paragraph_name = s;
  next = getNextLine();
  while ((next != null) && ( !next.startsWith("[")))
  {
    if (next.startsWith("#"))// skip the lines beginning with #.
    {
      next = getNextLine();
      continue;
    }
    ini = new iniElement(next);
    if ( !ini.error())
    {
      // put the property read in the hash table.
      tags.put(ini.tagName(), ini.valueName());
    }
    next = getNextLine();
  }
  if (tags.size() > 0)
    error_flag = false;
}

public iniParagraph(InputFile fl, String pname)
{
  f = fl;
  read_to_nextparagraph(pname);
}
```

```
  public iniParagraph(String pname)
  {
    // call this constructor to create a new paragraph
    tags = new Hashtable();
    paragraph_name = pname;
  }

  private String getNextLine()
  {
    line = f.readLine();
    while ((line != null) && (line.length() < 1))
      line = f.readLine();
    if (line == null)
      error_flag = true;
    else
      line = line.trim();
    return line;
  }

  public boolean error()
  {
    return error_flag;
  }

  public String nextLine()
  {
    return line;
  }
}
```

iniElement.java

```
package anova.util;

/** A private class used by IniFile */
public class iniElement
{
  private String tag, val;

  private boolean error_flag;

  // parses one element of an ini-file
  // into a tag and a value
  public iniElement(String s)
  {
    error_flag = false;
    int i = s.indexOf("=");
    if (i > 0)
    {
      tag = s.substring(0, i).trim().toLowerCase();
      val = s.substring(i + 1);
    }
    else
      error_flag = true;
  }

  public boolean error()
  {
    return error_flag;
  }

  public String tagName()
  {
    return tag;
  }

  public String valueName()
  {
    return val;
  }
}
```

InputFile.java

```
package anova.util;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;

public class InputFile
{
 private BufferedReader f = null;

 private boolean errflag;

 private String s = null;

 private String filename;

 public InputFile(String fname) throws IOException
 {
  errflag = false;

  // open file
  fname = fname.trim();
  f = new BufferedReader(new FileReader(fname));
  filename = fname;
 }

 public boolean checkErr()
 {
  return errflag;
 }

 public String read()
 {
  // read a single field up to a comma or end of line
  String ret = "";
  if (s == null)
  { // if no data in string
    s = readLine(); // read next line
  }
  if (s != null)
  { // if there is data
    s.trim(); // trim off blanks
    int i = s.indexOf(","); // find next comma
```

```
    if (i <= 0)
    {
      ret = s.trim(); // if no commas go to end of line
      s = null; // and null out stored string
    }
    else
    {
      ret = s.substring(0, i).trim(); // return left of comma
      s = s.substring(i + 1); // save right of comma
    }
  }
  else
    ret = null;
  return ret; // return string
}

public String readLine()
{
  // read in a line from the file
  s = null;
  try
  {
    s = f.readLine(); // could throw error
  }
  catch (IOException e)
  {
    errflag = true;
    System.out.println("File read error");
  }
  return s;
}

public void close()
{
  try
  {
    f.close(); // close file
  }
  catch (IOException e)
  {
    System.out.println("File close error");
    errflag = true;
  }
}

public String root()
```

```
  {
    int i = filename.lastIndexOf(File.separatorChar);
    String s = filename.substring(i + 1);
    i = s.indexOf(".");
    return s.substring(0, i);
  }
}
```

OutputFile.java

```
package anova.util;

import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;

// Output file class
public class OutputFile
{
  private BufferedWriter f;

  private PrintWriter p;

  private int tabcolumn;

  private int width;

  /**
   * Create output file
   *
   * @param complete
   *          path and name of file
   */
  public OutputFile(String filename) throws IOException
  {
    tabcolumn = 0;
    width = 0;
    filename = filename.trim();
    System.out.println(filename);
    f = new BufferedWriter(new FileWriter(filename));
    p = new PrintWriter(f);
  }

  /** insert a tab into the file */
  public void tab()
  {
    p.print("\t");
  }

  /**
   * insert spaces into the file
   *
   * @param the
```

```
     *          number of spaces to insert
     */
    public String space(int n)
    {
      StringBuffer sb = new StringBuffer(n);
      // put spaces into string buffer
      for (int i = 0; i < n; i++)
      {
        sb.insert(i, ' ');
      }
      return sb.toString();
    }

    /**
     * tab over to column
     *
     * @param the
     *          column to tab over to Inserts spaces to move up to tab column or
     *          starts a new line
     */
    public void tab(int tb)
    {
      if (tb > tabcolumn)
      {
        print(space(tb - tabcolumn));
      }
      else
        println("");
    }

    public void println(String s)
    {
      p.println(s);
      tabcolumn = 0;
    }

    public void println(int i)
    {
      p.println(i);
      tabcolumn = 0;
    }

    public void println(double d)
    {
      tabcolumn = 0;
      p.println(d);
```

```
  }

  public void print(String s)
  {
    p.print(s);
    tabcolumn += s.length();
  }

  public void print(int i)
  {
    String s = new Integer(i).toString();
    if (s.length() < width)
      print(space(width - s.length()));
    print(s);
  }

  public void print(float f)
  {
    String s = new Float(f).toString();
    print(s);
  }

  public void print(double d)
  {
    String s = new Double(d).toString();
    print(s);
  }

  public void close()
  {
    p.close();
  }

  public void finalize()
  {
    close();
  }
}
```

APPENDIX C

The following files can be used to test an installation of the ANOVA Java program to insure that it is operating properly.

TEST CONFIGURATION FILE

```
[config]
# Number of factors to consider in Anova.
NumOfFactors = 3
# The following must be enclosed in double quotes.
ColumnSeparator = ","
DataFile = sampleinput.txt
TwoWayOutputFile = SampleTwoWayAnovaOutput.txt
OneWayOutputFile = SampleOneWayAnovaOutput.txt
HeadersForOneWayAnova = Factor2

# Now describe each factor individually.
# A factor paragraph has the format Factor#, where # is substituted by an integer.
[Factor1]
Name = sample
Columns = 2
MaxNumOfIndex = 6

[Factor2]
Name = condition
Columns = 3
MaxNumOfIndex = 3
Enumerate = Heat   1        Cold   2        Control        3

[Factor3]
Name = time
Columns = 4
MaxNumOfIndex = 2
```

TEST INPUT FILE

AFFX-BioB-5_at,1,Control,1,470.90
AFFX-BioB-5_at,1,Control,2,455.50
AFFX-BioB-5_at,1,Heat,1,486.70
AFFX-BioB-5_at,1,Heat,2,464.40
AFFX-BioB-5_at,1,Cold,1,378.40
AFFX-BioB-5_at,1,Cold,2,380.00
AFFX-BioB-5_at,2,Control,1,370.40
AFFX-BioB-5_at,2,Control,2,399.30
AFFX-BioB-5_at,2,Heat,1,418.20
AFFX-BioB-5_at,2,Heat,2,454.70
AFFX-BioB-5_at,2,Cold,1,395.30
AFFX-BioB-5_at,2,Cold,2,425.40
AFFX-BioB-5_at,3,Control,1,393.40
AFFX-BioB-5_at,3,Control,2,392.70
AFFX-BioB-5_at,3,Heat,1,386.60
AFFX-BioB-5_at,3,Heat,2,428.60
AFFX-BioB-5_at,3,Cold,1,282.90
AFFX-BioB-5_at,3,Cold,2,331.80
AFFX-BioB-5_at,4,Control,1,347.80
AFFX-BioB-5_at,4,Control,2,389.80
AFFX-BioB-5_at,4,Heat,1,393.80
AFFX-BioB-5_at,4,Heat,2,443.60
AFFX-BioB-5_at,4,Cold,1,354.20
AFFX-BioB-5_at,4,Cold,2,432.30
AFFX-BioB-5_at,5,Control,1,339.50
AFFX-BioB-5_at,5,Control,2,373.10
AFFX-BioB-5_at,5,Heat,1,402.40
AFFX-BioB-5_at,5,Heat,2,524.00
AFFX-BioB-5_at,5,Cold,1,346.10
AFFX-BioB-5_at,5,Cold,2,360.60
AFFX-BioB-5_at,6,Control,1,354.50
AFFX-BioB-5_at,6,Control,2,314.40
AFFX-BioB-5_at,6,Heat,1,376.80
AFFX-BioB-5_at,6,Heat,2,1097.10
AFFX-BioB-5_at,6,Cold,1,484.50
AFFX-BioB-5_at,6,Cold,2,419.20
AFFX-BioB-M_at,1,Control,1,776.20
AFFX-BioB-M_at,1,Control,2,963.80
AFFX-BioB-M_at,1,Heat,1,734.30
AFFX-BioB-M_at,1,Heat,2,700.20
AFFX-BioB-M_at,1,Cold,1,630.00
AFFX-BioB-M_at,1,Cold,2,641.00
AFFX-BioB-M_at,2,Control,1,649.10
AFFX-BioB-M_at,2,Control,2,787.90

AFFX-BioB-M_at,2,Heat,1,627.60
AFFX-BioB-M_at,2,Heat,2,528.20
AFFX-BioB-M_at,2,Cold,1,751.40
AFFX-BioB-M_at,2,Cold,2,501.10
AFFX-BioB-M_at,3,Control,1,589.00
AFFX-BioB-M_at,3,Control,2,655.50
AFFX-BioB-M_at,3,Heat,1,720.30
AFFX-BioB-M_at,3,Heat,2,534.00
AFFX-BioB-M_at,3,Cold,1,674.80
AFFX-BioB-M_at,3,Cold,2,552.00
AFFX-BioB-M_at,4,Control,1,400.70
AFFX-BioB-M_at,4,Control,2,479.40
AFFX-BioB-M_at,4,Heat,1,520.60
AFFX-BioB-M_at,4,Heat,2,616.80
AFFX-BioB-M_at,4,Cold,1,423.60
AFFX-BioB-M_at,4,Cold,2,619.40
AFFX-BioB-M_at,5,Control,1,431.50
AFFX-BioB-M_at,5,Control,2,530.00
AFFX-BioB-M_at,5,Heat,1,578.40
AFFX-BioB-M_at,5,Heat,2,661.10
AFFX-BioB-M_at,5,Cold,1,450.80
AFFX-BioB-M_at,5,Cold,2,462.60
AFFX-BioB-M_at,6,Control,1,535.40
AFFX-BioB-M_at,6,Control,2,389.90
AFFX-BioB-M_at,6,Heat,1,529.90
AFFX-BioB-M_at,6,Heat,2,1789.60
AFFX-BioB-M_at,6,Cold,1,558.20
AFFX-BioB-M_at,6,Cold,2,656.60
AFFX-BioB-3_at,1,Control,1,405.00
AFFX-BioB-3_at,1,Control,2,422.10
AFFX-BioB-3_at,1,Heat,1,475.60
AFFX-BioB-3_at,1,Heat,2,537.20
AFFX-BioB-3_at,1,Cold,1,396.50
AFFX-BioB-3_at,1,Cold,2,444.00
AFFX-BioB-3_at,2,Control,1,202.10
AFFX-BioB-3_at,2,Control,2,436.90
AFFX-BioB-3_at,2,Heat,1,364.70
AFFX-BioB-3_at,2,Heat,2,365.80
AFFX-BioB-3_at,2,Cold,1,393.10
AFFX-BioB-3_at,2,Cold,2,274.30
AFFX-BioB-3_at,3,Control,1,308.80
AFFX-BioB-3_at,3,Control,2,425.60
AFFX-BioB-3_at,3,Heat,1,516.90
AFFX-BioB-3_at,3,Heat,2,368.60
AFFX-BioB-3_at,3,Cold,1,384.70
AFFX-BioB-3_at,3,Cold,2,320.50

AFFX-BioB-3_at,4,Control,1,249.00
AFFX-BioB-3_at,4,Control,2,265.60
AFFX-BioB-3_at,4,Heat,1,272.60
AFFX-BioB-3_at,4,Heat,2,335.40
AFFX-BioB-3_at,4,Cold,1,255.00
AFFX-BioB-3_at,4,Cold,2,290.40
AFFX-BioB-3_at,5,Control,1,293.80
AFFX-BioB-3_at,5,Control,2,252.30
AFFX-BioB-3_at,5,Heat,1,304.50
AFFX-BioB-3_at,5,Heat,2,400.50
AFFX-BioB-3_at,5,Cold,1,260.80
AFFX-BioB-3_at,5,Cold,2,266.20
AFFX-BioB-3_at,6,Control,1,286.30
AFFX-BioB-3_at,6,Control,2,278.70
AFFX-BioB-3_at,6,Heat,1,303.80
AFFX-BioB-3_at,6,Heat,2,1142.50
AFFX-BioB-3_at,6,Cold,1,307.60
AFFX-BioB-3_at,6,Cold,2,312.00
AFFX-BioC-5_at,1,Control,1,1036.40
AFFX-BioC-5_at,1,Control,2,1105.70
AFFX-BioC-5_at,1,Heat,1,1051.80
AFFX-BioC-5_at,1,Heat,2,982.60
AFFX-BioC-5_at,1,Cold,1,1066.50
AFFX-BioC-5_at,1,Cold,2,1170.70
AFFX-BioC-5_at,2,Control,1,907.40
AFFX-BioC-5_at,2,Control,2,986.50
AFFX-BioC-5_at,2,Heat,1,1144.60
AFFX-BioC-5_at,2,Heat,2,1067.00
AFFX-BioC-5_at,2,Cold,1,1113.90
AFFX-BioC-5_at,2,Cold,2,909.20
AFFX-BioC-5_at,3,Control,1,918.30
AFFX-BioC-5_at,3,Control,2,954.30
AFFX-BioC-5_at,3,Heat,1,1032.80
AFFX-BioC-5_at,3,Heat,2,864.50
AFFX-BioC-5_at,3,Cold,1,950.50
AFFX-BioC-5_at,3,Cold,2,803.30
AFFX-BioC-5_at,4,Control,1,872.50
AFFX-BioC-5_at,4,Control,2,923.40
AFFX-BioC-5_at,4,Heat,1,939.60
AFFX-BioC-5_at,4,Heat,2,1033.90
AFFX-BioC-5_at,4,Cold,1,857.80
AFFX-BioC-5_at,4,Cold,2,986.20
AFFX-BioC-5_at,5,Control,1,1067.30
AFFX-BioC-5_at,5,Control,2,1071.00
AFFX-BioC-5_at,5,Heat,1,1122.30
AFFX-BioC-5_at,5,Heat,2,1226.00

AFFX-BioC-5_at,5,Cold,1,911.80
AFFX-BioC-5_at,5,Cold,2,1010.70
AFFX-BioC-5_at,6,Control,1,936.10
AFFX-BioC-5_at,6,Control,2,974.40
AFFX-BioC-5_at,6,Heat,1,943.00
AFFX-BioC-5_at,6,Heat,2,2778.90
AFFX-BioC-5_at,6,Cold,1,1150.00
AFFX-BioC-5_at,6,Cold,2,1222.30
AFFX-BioC-3_at,1,Control,1,989.10
AFFX-BioC-3_at,1,Control,2,870.20
AFFX-BioC-3_at,1,Heat,1,910.80
AFFX-BioC-3_at,1,Heat,2,879.90
AFFX-BioC-3_at,1,Cold,1,820.10
AFFX-BioC-3_at,1,Cold,2,816.60
AFFX-BioC-3_at,2,Control,1,668.00
AFFX-BioC-3_at,2,Control,2,749.70
AFFX-BioC-3_at,2,Heat,1,864.30
AFFX-BioC-3_at,2,Heat,2,759.30
AFFX-BioC-3_at,2,Cold,1,824.00
AFFX-BioC-3_at,2,Cold,2,731.30
AFFX-BioC-3_at,3,Control,1,575.10
AFFX-BioC-3_at,3,Control,2,741.90
AFFX-BioC-3_at,3,Heat,1,801.40
AFFX-BioC-3_at,3,Heat,2,699.80
AFFX-BioC-3_at,3,Cold,1,758.40
AFFX-BioC-3_at,3,Cold,2,641.60
AFFX-BioC-3_at,4,Control,1,731.50
AFFX-BioC-3_at,4,Control,2,734.40
AFFX-BioC-3_at,4,Heat,1,739.00
AFFX-BioC-3_at,4,Heat,2,788.30
AFFX-BioC-3_at,4,Cold,1,696.30
AFFX-BioC-3_at,4,Cold,2,771.10
AFFX-BioC-3_at,5,Control,1,851.50
AFFX-BioC-3_at,5,Control,2,838.10
AFFX-BioC-3_at,5,Heat,1,804.50
AFFX-BioC-3_at,5,Heat,2,1014.00
AFFX-BioC-3_at,5,Cold,1,692.00
AFFX-BioC-3_at,5,Cold,2,810.70
AFFX-BioC-3_at,6,Control,1,752.20
AFFX-BioC-3_at,6,Control,2,827.40
AFFX-BioC-3_at,6,Heat,1,753.40
AFFX-BioC-3_at,6,Heat,2,2810.00
AFFX-BioC-3_at,6,Cold,1,843.90
AFFX-BioC-3_at,6,Cold,2,770.60
AFFX-BioDn-5_at,1,Control,1,1013.50
AFFX-BioDn-5_at,1,Control,2,1263.80

AFFX-BioDn-5_at,1,Heat,1,1331.20
AFFX-BioDn-5_at,1,Heat,2,1157.70
AFFX-BioDn-5_at,1,Cold,1,1008.00
AFFX-BioDn-5_at,1,Cold,2,935.00
AFFX-BioDn-5_at,2,Control,1,867.00
AFFX-BioDn-5_at,2,Control,2,1007.60
AFFX-BioDn-5_at,2,Heat,1,1132.10
AFFX-BioDn-5_at,2,Heat,2,1013.60
AFFX-BioDn-5_at,2,Cold,1,927.50
AFFX-BioDn-5_at,2,Cold,2,966.40
AFFX-BioDn-5_at,3,Control,1,885.50
AFFX-BioDn-5_at,3,Control,2,881.40
AFFX-BioDn-5_at,3,Heat,1,812.20
AFFX-BioDn-5_at,3,Heat,2,950.00
AFFX-BioDn-5_at,3,Cold,1,885.90
AFFX-BioDn-5_at,3,Cold,2,728.20
AFFX-BioDn-5_at,4,Control,1,949.10
AFFX-BioDn-5_at,4,Control,2,1097.60
AFFX-BioDn-5_at,4,Heat,1,1040.70
AFFX-BioDn-5_at,4,Heat,2,1344.60
AFFX-BioDn-5_at,4,Cold,1,929.30
AFFX-BioDn-5_at,4,Cold,2,1135.10
AFFX-BioDn-5_at,5,Control,1,1247.80
AFFX-BioDn-5_at,5,Control,2,1068.40
AFFX-BioDn-5_at,5,Heat,1,1262.20
AFFX-BioDn-5_at,5,Heat,2,1570.90
AFFX-BioDn-5_at,5,Cold,1,1172.80
AFFX-BioDn-5_at,5,Cold,2,1060.80
AFFX-BioDn-5_at,6,Control,1,1053.60
AFFX-BioDn-5_at,6,Control,2,1086.10
AFFX-BioDn-5_at,6,Heat,1,1146.90
AFFX-BioDn-5_at,6,Heat,2,3057.50
AFFX-BioDn-5_at,6,Cold,1,1186.40
AFFX-BioDn-5_at,6,Cold,2,1186.20
AFFX-BioDn-3_at,1,Control,1,7207.30
AFFX-BioDn-3_at,1,Control,2,9109.50
AFFX-BioDn-3_at,1,Heat,1,7460.60
AFFX-BioDn-3_at,1,Heat,2,8004.00
AFFX-BioDn-3_at,1,Cold,1,6797.00
AFFX-BioDn-3_at,1,Cold,2,7549.40
AFFX-BioDn-3_at,2,Control,1,6300.50
AFFX-BioDn-3_at,2,Control,2,6465.00
AFFX-BioDn-3_at,2,Heat,1,7307.40
AFFX-BioDn-3_at,2,Heat,2,6776.60
AFFX-BioDn-3_at,2,Cold,1,6472.10
AFFX-BioDn-3_at,2,Cold,2,6229.00

AFFX-BioDn-3_at,3,Control,1,6437.40
AFFX-BioDn-3_at,3,Control,2,6234.80
AFFX-BioDn-3_at,3,Heat,1,6924.40
AFFX-BioDn-3_at,3,Heat,2,6761.00
AFFX-BioDn-3_at,3,Cold,1,6080.10
AFFX-BioDn-3_at,3,Cold,2,6227.30
AFFX-BioDn-3_at,4,Control,1,6211.50
AFFX-BioDn-3_at,4,Control,2,5820.80
AFFX-BioDn-3_at,4,Heat,1,6362.70
AFFX-BioDn-3_at,4,Heat,2,8059.10
AFFX-BioDn-3_at,4,Cold,1,6298.40
AFFX-BioDn-3_at,4,Cold,2,6755.40
AFFX-BioDn-3_at,5,Control,1,6650.00
AFFX-BioDn-3_at,5,Control,2,6296.00
AFFX-BioDn-3_at,5,Heat,1,7538.70
AFFX-BioDn-3_at,5,Heat,2,9636.70
AFFX-BioDn-3_at,5,Cold,1,6252.10
AFFX-BioDn-3_at,5,Cold,2,6886.40
AFFX-BioDn-3_at,6,Control,1,7114.50
AFFX-BioDn-3_at,6,Control,2,6865.70
AFFX-BioDn-3_at,6,Heat,1,8175.40
AFFX-BioDn-3_at,6,Heat,2,22718.70
AFFX-BioDn-3_at,6,Cold,1,8222.10
AFFX-BioDn-3_at,6,Cold,2,7978.20
AFFX-CreX-5_at,1,Control,1,10349.20
AFFX-CreX-5_at,1,Control,2,10896.80
AFFX-CreX-5_at,1,Heat,1,10961.60
AFFX-CreX-5_at,1,Heat,2,10869.30
AFFX-CreX-5_at,1,Cold,1,10200.50
AFFX-CreX-5_at,1,Cold,2,9888.20
AFFX-CreX-5_at,2,Control,1,8218.50
AFFX-CreX-5_at,2,Control,2,11730.00
AFFX-CreX-5_at,2,Heat,1,10366.00
AFFX-CreX-5_at,2,Heat,2,10479.70
AFFX-CreX-5_at,2,Cold,1,10053.30
AFFX-CreX-5_at,2,Cold,2,10261.80
AFFX-CreX-5_at,3,Control,1,9099.40
AFFX-CreX-5_at,3,Control,2,10479.70
AFFX-CreX-5_at,3,Heat,1,10265.50
AFFX-CreX-5_at,3,Heat,2,9779.50
AFFX-CreX-5_at,3,Cold,1,8689.50
AFFX-CreX-5_at,3,Cold,2,10807.20
AFFX-CreX-5_at,4,Control,1,8572.30
AFFX-CreX-5_at,4,Control,2,8808.80
AFFX-CreX-5_at,4,Heat,1,9909.80
AFFX-CreX-5_at,4,Heat,2,11175.80

AFFX-CreX-5_at,4,Cold,1,8380.30
AFFX-CreX-5_at,4,Cold,2,10343.70
AFFX-CreX-5_at,5,Control,1,10793.60
AFFX-CreX-5_at,5,Control,2,9764.40
AFFX-CreX-5_at,5,Heat,1,11415.80
AFFX-CreX-5_at,5,Heat,2,14937.50
AFFX-CreX-5_at,5,Cold,1,9218.20
AFFX-CreX-5_at,5,Cold,2,11477.30
AFFX-CreX-5_at,6,Control,1,10291.30
AFFX-CreX-5_at,6,Control,2,9882.50
AFFX-CreX-5_at,6,Heat,1,10594.00
AFFX-CreX-5_at,6,Heat,2,31006.60
AFFX-CreX-5_at,6,Cold,1,10829.10
AFFX-CreX-5_at,6,Cold,2,11607.30
AFFX-CreX-3_at,1,Control,1,14407.10
AFFX-CreX-3_at,1,Control,2,16371.70
AFFX-CreX-3_at,1,Heat,1,14402.70
AFFX-CreX-3_at,1,Heat,2,15447.60
AFFX-CreX-3_at,1,Cold,1,14135.00
AFFX-CreX-3_at,1,Cold,2,15241.30
AFFX-CreX-3_at,2,Control,1,11974.20
AFFX-CreX-3_at,2,Control,2,15715.90
AFFX-CreX-3_at,2,Heat,1,15349.30
AFFX-CreX-3_at,2,Heat,2,16210.30
AFFX-CreX-3_at,2,Cold,1,14645.00
AFFX-CreX-3_at,2,Cold,2,14767.40
AFFX-CreX-3_at,3,Control,1,12924.20
AFFX-CreX-3_at,3,Control,2,14047.80
AFFX-CreX-3_at,3,Heat,1,15006.80
AFFX-CreX-3_at,3,Heat,2,14172.70
AFFX-CreX-3_at,3,Cold,1,12170.40
AFFX-CreX-3_at,3,Cold,2,14946.40
AFFX-CreX-3_at,4,Control,1,11179.20
AFFX-CreX-3_at,4,Control,2,11960.40
AFFX-CreX-3_at,4,Heat,1,12972.60
AFFX-CreX-3_at,4,Heat,2,14237.70
AFFX-CreX-3_at,4,Cold,1,11492.30
AFFX-CreX-3_at,4,Cold,2,13275.60
AFFX-CreX-3_at,5,Control,1,15797.40
AFFX-CreX-3_at,5,Control,2,15410.30
AFFX-CreX-3_at,5,Heat,1,16493.20
AFFX-CreX-3_at,5,Heat,2,20345.00
AFFX-CreX-3_at,5,Cold,1,12950.00
AFFX-CreX-3_at,5,Cold,2,16059.10
AFFX-CreX-3_at,6,Control,1,13583.80
AFFX-CreX-3_at,6,Control,2,14567.80

AFFX-CreX-3_at,6,Heat,1,14343.30
AFFX-CreX-3_at,6,Heat,2,41726.10
AFFX-CreX-3_at,6,Cold,1,15294.30
AFFX-CreX-3_at,6,Cold,2,14518.30
AFFX-DapX-5_at,1,Control,1,14.30
AFFX-DapX-5_at,1,Control,2,61.50
AFFX-DapX-5_at,1,Heat,1,24.70
AFFX-DapX-5_at,1,Heat,2,72.30
AFFX-DapX-5_at,1,Cold,1,45.50
AFFX-DapX-5_at,1,Cold,2,90.00
AFFX-DapX-5_at,2,Control,1,93.60
AFFX-DapX-5_at,2,Control,2,74.40
AFFX-DapX-5_at,2,Heat,1,37.80
AFFX-DapX-5_at,2,Heat,2,86.20
AFFX-DapX-5_at,2,Cold,1,56.10
AFFX-DapX-5_at,2,Cold,2,33.10
AFFX-DapX-5_at,3,Control,1,9.90
AFFX-DapX-5_at,3,Control,2,42.00
AFFX-DapX-5_at,3,Heat,1,39.50
AFFX-DapX-5_at,3,Heat,2,23.50
AFFX-DapX-5_at,3,Cold,1,3.90
AFFX-DapX-5_at,3,Cold,2,53.80
AFFX-DapX-5_at,4,Control,1,80.00
AFFX-DapX-5_at,4,Control,2,8.00
AFFX-DapX-5_at,4,Heat,1,56.50
AFFX-DapX-5_at,4,Heat,2,41.10
AFFX-DapX-5_at,4,Cold,1,76.30
AFFX-DapX-5_at,4,Cold,2,65.10
AFFX-DapX-5_at,5,Control,1,36.00
AFFX-DapX-5_at,5,Control,2,26.10
AFFX-DapX-5_at,5,Heat,1,9.80
AFFX-DapX-5_at,5,Heat,2,36.50
AFFX-DapX-5_at,5,Cold,1,27.60
AFFX-DapX-5_at,5,Cold,2,12.80
AFFX-DapX-5_at,6,Control,1,30.40
AFFX-DapX-5_at,6,Control,2,14.50
AFFX-DapX-5_at,6,Heat,1,38.00
AFFX-DapX-5_at,6,Heat,2,15.80
AFFX-DapX-5_at,6,Cold,1,49.30
AFFX-DapX-5_at,6,Cold,2,7.80
AFFX-DapX-M_at,1,Control,1,66.30
AFFX-DapX-M_at,1,Control,2,116.80
AFFX-DapX-M_at,1,Heat,1,77.60
AFFX-DapX-M_at,1,Heat,2,45.80
AFFX-DapX-M_at,1,Cold,1,113.50
AFFX-DapX-M_at,1,Cold,2,65.40

AFFX-DapX-M_at,2,Control,1,133.40
AFFX-DapX-M_at,2,Control,2,89.70
AFFX-DapX-M_at,2,Heat,1,56.30
AFFX-DapX-M_at,2,Heat,2,53.90
AFFX-DapX-M_at,2,Cold,1,60.10
AFFX-DapX-M_at,2,Cold,2,29.30
AFFX-DapX-M_at,3,Control,1,56.70
AFFX-DapX-M_at,3,Control,2,142.60
AFFX-DapX-M_at,3,Heat,1,54.00
AFFX-DapX-M_at,3,Heat,2,86.70
AFFX-DapX-M_at,3,Cold,1,113.40
AFFX-DapX-M_at,3,Cold,2,10.40
AFFX-DapX-M_at,4,Control,1,92.30
AFFX-DapX-M_at,4,Control,2,46.10
AFFX-DapX-M_at,4,Heat,1,93.00
AFFX-DapX-M_at,4,Heat,2,81.00
AFFX-DapX-M_at,4,Cold,1,109.70
AFFX-DapX-M_at,4,Cold,2,35.00
AFFX-DapX-M_at,5,Control,1,66.20
AFFX-DapX-M_at,5,Control,2,66.30
AFFX-DapX-M_at,5,Heat,1,9.80
AFFX-DapX-M_at,5,Heat,2,72.30
AFFX-DapX-M_at,5,Cold,1,102.10
AFFX-DapX-M_at,5,Cold,2,101.50
AFFX-DapX-M_at,6,Control,1,78.30
AFFX-DapX-M_at,6,Control,2,57.90
AFFX-DapX-M_at,6,Heat,1,127.80
AFFX-DapX-M_at,6,Heat,2,221.80
AFFX-DapX-M_at,6,Cold,1,94.20
AFFX-DapX-M_at,6,Cold,2,99.10
AFFX-DapX-3_at,1,Control,1,35.50
AFFX-DapX-3_at,1,Control,2,11.00
AFFX-DapX-3_at,1,Heat,1,7.80
AFFX-DapX-3_at,1,Heat,2,14.10
AFFX-DapX-3_at,1,Cold,1,11.20
AFFX-DapX-3_at,1,Cold,2,11.00
AFFX-DapX-3_at,2,Control,1,11.80
AFFX-DapX-3_at,2,Control,2,17.80
AFFX-DapX-3_at,2,Heat,1,9.00
AFFX-DapX-3_at,2,Heat,2,12.30
AFFX-DapX-3_at,2,Cold,1,8.50
AFFX-DapX-3_at,2,Cold,2,20.40
AFFX-DapX-3_at,3,Control,1,15.50
AFFX-DapX-3_at,3,Control,2,12.40
AFFX-DapX-3_at,3,Heat,1,9.50
AFFX-DapX-3_at,3,Heat,2,7.30

AFFX-DapX-3_at,3,Cold,1,5.60
AFFX-DapX-3_at,3,Cold,2,14.90
AFFX-DapX-3_at,4,Control,1,11.60
AFFX-DapX-3_at,4,Control,2,7.60
AFFX-DapX-3_at,4,Heat,1,9.40
AFFX-DapX-3_at,4,Heat,2,11.40
AFFX-DapX-3_at,4,Cold,1,12.10
AFFX-DapX-3_at,4,Cold,2,10.80
AFFX-DapX-3_at,5,Control,1,8.20
AFFX-DapX-3_at,5,Control,2,8.70
AFFX-DapX-3_at,5,Heat,1,6.60
AFFX-DapX-3_at,5,Heat,2,11.10
AFFX-DapX-3_at,5,Cold,1,10.90
AFFX-DapX-3_at,5,Cold,2,12.90
AFFX-DapX-3_at,6,Control,1,12.70
AFFX-DapX-3_at,6,Control,2,7.60
AFFX-DapX-3_at,6,Heat,1,5.80
AFFX-DapX-3_at,6,Heat,2,18.70
AFFX-DapX-3_at,6,Cold,1,13.40
AFFX-DapX-3_at,6,Cold,2,9.00
AFFX-LysX-5_at,1,Control,1,5.30
AFFX-LysX-5_at,1,Control,2,29.70
AFFX-LysX-5_at,1,Heat,1,6.40
AFFX-LysX-5_at,1,Heat,2,6.30
AFFX-LysX-5_at,1,Cold,1,28.90
AFFX-LysX-5_at,1,Cold,2,57.20
AFFX-LysX-5_at,2,Control,1,44.10
AFFX-LysX-5_at,2,Control,2,32.10
AFFX-LysX-5_at,2,Heat,1,8.70
AFFX-LysX-5_at,2,Heat,2,2.50
AFFX-LysX-5_at,2,Cold,1,44.30
AFFX-LysX-5_at,2,Cold,2,4.20
AFFX-LysX-5_at,3,Control,1,14.20
AFFX-LysX-5_at,3,Control,2,30.70
AFFX-LysX-5_at,3,Heat,1,25.70
AFFX-LysX-5_at,3,Heat,2,12.30
AFFX-LysX-5_at,3,Cold,1,20.40
AFFX-LysX-5_at,3,Cold,2,9.50
AFFX-LysX-5_at,4,Control,1,7.00
AFFX-LysX-5_at,4,Control,2,8.00
AFFX-LysX-5_at,4,Heat,1,22.60
AFFX-LysX-5_at,4,Heat,2,6.40
AFFX-LysX-5_at,4,Cold,1,5.70
AFFX-LysX-5_at,4,Cold,2,16.30
AFFX-LysX-5_at,5,Control,1,20.50
AFFX-LysX-5_at,5,Control,2,21.10

AFFX-LysX-5_at,5,Heat,1,12.40
AFFX-LysX-5_at,5,Heat,2,9.80
AFFX-LysX-5_at,5,Cold,1,6.10
AFFX-LysX-5_at,5,Cold,2,8.60
AFFX-LysX-5_at,6,Control,1,11.00
AFFX-LysX-5_at,6,Control,2,6.80
AFFX-LysX-5_at,6,Heat,1,11.40
AFFX-LysX-5_at,6,Heat,2,60.70
AFFX-LysX-5_at,6,Cold,1,11.90
AFFX-LysX-5_at,6,Cold,2,9.70
AFFX-LysX-M_at,1,Control,1,75.30
AFFX-LysX-M_at,1,Control,2,61.90
AFFX-LysX-M_at,1,Heat,1,53.70
AFFX-LysX-M_at,1,Heat,2,27.40
AFFX-LysX-M_at,1,Cold,1,54.90
AFFX-LysX-M_at,1,Cold,2,59.50
AFFX-LysX-M_at,2,Control,1,78.70
AFFX-LysX-M_at,2,Control,2,33.30
AFFX-LysX-M_at,2,Heat,1,24.30
AFFX-LysX-M_at,2,Heat,2,25.80
AFFX-LysX-M_at,2,Cold,1,23.40
AFFX-LysX-M_at,2,Cold,2,9.90
AFFX-LysX-M_at,3,Control,1,42.30
AFFX-LysX-M_at,3,Control,2,80.70
AFFX-LysX-M_at,3,Heat,1,30.80
AFFX-LysX-M_at,3,Heat,2,98.10
AFFX-LysX-M_at,3,Cold,1,20.30
AFFX-LysX-M_at,3,Cold,2,19.70
AFFX-LysX-M_at,4,Control,1,19.80
AFFX-LysX-M_at,4,Control,2,24.20
AFFX-LysX-M_at,4,Heat,1,27.80
AFFX-LysX-M_at,4,Heat,2,50.50
AFFX-LysX-M_at,4,Cold,1,20.20
AFFX-LysX-M_at,4,Cold,2,23.10
AFFX-LysX-M_at,5,Control,1,23.90
AFFX-LysX-M_at,5,Control,2,35.00
AFFX-LysX-M_at,5,Heat,1,15.70
AFFX-LysX-M_at,5,Heat,2,41.70
AFFX-LysX-M_at,5,Cold,1,44.60
AFFX-LysX-M_at,5,Cold,2,26.30
AFFX-LysX-M_at,6,Control,1,20.10
AFFX-LysX-M_at,6,Control,2,43.00
AFFX-LysX-M_at,6,Heat,1,33.70
AFFX-LysX-M_at,6,Heat,2,144.40
AFFX-LysX-M_at,6,Cold,1,78.20
AFFX-LysX-M_at,6,Cold,2,85.30

AFFX-LysX-3_at,1,Control,1,8.80
AFFX-LysX-3_at,1,Control,2,46.30
AFFX-LysX-3_at,1,Heat,1,27.10
AFFX-LysX-3_at,1,Heat,2,38.90
AFFX-LysX-3_at,1,Cold,1,6.30
AFFX-LysX-3_at,1,Cold,2,61.50
AFFX-LysX-3_at,2,Control,1,50.70
AFFX-LysX-3_at,2,Control,2,9.70
AFFX-LysX-3_at,2,Heat,1,22.70
AFFX-LysX-3_at,2,Heat,2,8.50
AFFX-LysX-3_at,2,Cold,1,4.10
AFFX-LysX-3_at,2,Cold,2,4.70
AFFX-LysX-3_at,3,Control,1,2.10
AFFX-LysX-3_at,3,Control,2,7.50
AFFX-LysX-3_at,3,Heat,1,36.70
AFFX-LysX-3_at,3,Heat,2,8.30
AFFX-LysX-3_at,3,Cold,1,16.60
AFFX-LysX-3_at,3,Cold,2,27.40
AFFX-LysX-3_at,4,Control,1,12.20
AFFX-LysX-3_at,4,Control,2,10.70
AFFX-LysX-3_at,4,Heat,1,7.20
AFFX-LysX-3_at,4,Heat,2,56.80
AFFX-LysX-3_at,4,Cold,1,44.40
AFFX-LysX-3_at,4,Cold,2,30.50
AFFX-LysX-3_at,5,Control,1,19.90
AFFX-LysX-3_at,5,Control,2,6.30
AFFX-LysX-3_at,5,Heat,1,2.50
AFFX-LysX-3_at,5,Heat,2,15.40
AFFX-LysX-3_at,5,Cold,1,4.80
AFFX-LysX-3_at,5,Cold,2,27.10
AFFX-LysX-3_at,6,Control,1,36.00
AFFX-LysX-3_at,6,Control,2,11.60
AFFX-LysX-3_at,6,Heat,1,20.60
AFFX-LysX-3_at,6,Heat,2,39.20
AFFX-LysX-3_at,6,Cold,1,28.00
AFFX-LysX-3_at,6,Cold,2,8.00
AFFX-PheX-5_at,1,Control,1,8.10
AFFX-PheX-5_at,1,Control,2,5.10
AFFX-PheX-5_at,1,Heat,1,9.70
AFFX-PheX-5_at,1,Heat,2,6.50
AFFX-PheX-5_at,1,Cold,1,5.10
AFFX-PheX-5_at,1,Cold,2,13.30
AFFX-PheX-5_at,2,Control,1,9.10
AFFX-PheX-5_at,2,Control,2,10.50
AFFX-PheX-5_at,2,Heat,1,12.50
AFFX-PheX-5_at,2,Heat,2,10.70

AFFX-PheX-5_at,2,Cold,1,5.70
AFFX-PheX-5_at,2,Cold,2,5.90
AFFX-PheX-5_at,3,Control,1,8.80
AFFX-PheX-5_at,3,Control,2,11.30
AFFX-PheX-5_at,3,Heat,1,6.10
AFFX-PheX-5_at,3,Heat,2,6.40
AFFX-PheX-5_at,3,Cold,1,11.50
AFFX-PheX-5_at,3,Cold,2,6.20
AFFX-PheX-5_at,4,Control,1,4.90
AFFX-PheX-5_at,4,Control,2,3.40
AFFX-PheX-5_at,4,Heat,1,6.90
AFFX-PheX-5_at,4,Heat,2,9.00
AFFX-PheX-5_at,4,Cold,1,7.20
AFFX-PheX-5_at,4,Cold,2,7.80
AFFX-PheX-5_at,5,Control,1,5.00
AFFX-PheX-5_at,5,Control,2,7.30
AFFX-PheX-5_at,5,Heat,1,6.60
AFFX-PheX-5_at,5,Heat,2,5.10
AFFX-PheX-5_at,5,Cold,1,9.00
AFFX-PheX-5_at,5,Cold,2,6.20
AFFX-PheX-5_at,6,Control,1,7.40
AFFX-PheX-5_at,6,Control,2,5.10
AFFX-PheX-5_at,6,Heat,1,7.30
AFFX-PheX-5_at,6,Heat,2,9.40
AFFX-PheX-5_at,6,Cold,1,7.70
AFFX-PheX-5_at,6,Cold,2,13.20
AFFX-PheX-M_at,1,Control,1,8.20
AFFX-PheX-M_at,1,Control,2,11.60
AFFX-PheX-M_at,1,Heat,1,8.80
AFFX-PheX-M_at,1,Heat,2,12.30
AFFX-PheX-M_at,1,Cold,1,12.60
AFFX-PheX-M_at,1,Cold,2,16.80
AFFX-PheX-M_at,2,Control,1,17.90
AFFX-PheX-M_at,2,Control,2,8.70
AFFX-PheX-M_at,2,Heat,1,5.90
AFFX-PheX-M_at,2,Heat,2,16.20
AFFX-PheX-M_at,2,Cold,1,23.50
AFFX-PheX-M_at,2,Cold,2,12.10
AFFX-PheX-M_at,3,Control,1,8.90
AFFX-PheX-M_at,3,Control,2,5.80
AFFX-PheX-M_at,3,Heat,1,8.60
AFFX-PheX-M_at,3,Heat,2,10.10
AFFX-PheX-M_at,3,Cold,1,4.30
AFFX-PheX-M_at,3,Cold,2,8.10
AFFX-PheX-M_at,4,Control,1,10.70
AFFX-PheX-M_at,4,Control,2,6.70

AFFX-PheX-M_at,4,Heat,1,9.80
AFFX-PheX-M_at,4,Heat,2,9.10
AFFX-PheX-M_at,4,Cold,1,8.40
AFFX-PheX-M_at,4,Cold,2,4.60
AFFX-PheX-M_at,5,Control,1,4.90
AFFX-PheX-M_at,5,Control,2,8.40
AFFX-PheX-M_at,5,Heat,1,6.00
AFFX-PheX-M_at,5,Heat,2,5.90
AFFX-PheX-M_at,5,Cold,1,5.20
AFFX-PheX-M_at,5,Cold,2,6.90
AFFX-PheX-M_at,6,Control,1,7.60
AFFX-PheX-M_at,6,Control,2,7.70
AFFX-PheX-M_at,6,Heat,1,17.90
AFFX-PheX-M_at,6,Heat,2,28.20
AFFX-PheX-M_at,6,Cold,1,27.00
AFFX-PheX-M_at,6,Cold,2,13.10
AFFX-PheX-3_at,1,Control,1,113.50
AFFX-PheX-3_at,1,Control,2,28.80
AFFX-PheX-3_at,1,Heat,1,65.70
AFFX-PheX-3_at,1,Heat,2,150.70
AFFX-PheX-3_at,1,Cold,1,73.20
AFFX-PheX-3_at,1,Cold,2,56.90
AFFX-PheX-3_at,2,Control,1,43.80
AFFX-PheX-3_at,2,Control,2,88.80
AFFX-PheX-3_at,2,Heat,1,36.50
AFFX-PheX-3_at,2,Heat,2,82.80
AFFX-PheX-3_at,2,Cold,1,98.90
AFFX-PheX-3_at,2,Cold,2,36.20
AFFX-PheX-3_at,3,Control,1,69.80
AFFX-PheX-3_at,3,Control,2,89.90
AFFX-PheX-3_at,3,Heat,1,95.60
AFFX-PheX-3_at,3,Heat,2,61.90
AFFX-PheX-3_at,3,Cold,1,36.60
AFFX-PheX-3_at,3,Cold,2,40.40
AFFX-PheX-3_at,4,Control,1,68.90
AFFX-PheX-3_at,4,Control,2,37.00
AFFX-PheX-3_at,4,Heat,1,46.90
AFFX-PheX-3_at,4,Heat,2,109.90
AFFX-PheX-3_at,4,Cold,1,58.40
AFFX-PheX-3_at,4,Cold,2,91.50
AFFX-PheX-3_at,5,Control,1,107.80
AFFX-PheX-3_at,5,Control,2,32.50
AFFX-PheX-3_at,5,Heat,1,52.70
AFFX-PheX-3_at,5,Heat,2,37.60
AFFX-PheX-3_at,5,Cold,1,102.90
AFFX-PheX-3_at,5,Cold,2,118.40

AFFX-PheX-3_at,6,Control,1,85.70
AFFX-PheX-3_at,6,Control,2,99.50
AFFX-PheX-3_at,6,Heat,1,78.40
AFFX-PheX-3_at,6,Heat,2,241.00
AFFX-PheX-3_at,6,Cold,1,75.10
AFFX-PheX-3_at,6,Cold,2,79.70
AFFX-ThrX-5_at,1,Control,1,82.30
AFFX-ThrX-5_at,1,Control,2,19.30
AFFX-ThrX-5_at,1,Heat,1,23.70
AFFX-ThrX-5_at,1,Heat,2,26.40
AFFX-ThrX-5_at,1,Cold,1,42.90
AFFX-ThrX-5_at,1,Cold,2,22.90
AFFX-ThrX-5_at,2,Control,1,26.20
AFFX-ThrX-5_at,2,Control,2,34.30
AFFX-ThrX-5_at,2,Heat,1,29.00
AFFX-ThrX-5_at,2,Heat,2,31.00
AFFX-ThrX-5_at,2,Cold,1,30.40
AFFX-ThrX-5_at,2,Cold,2,19.30
AFFX-ThrX-5_at,3,Control,1,17.30
AFFX-ThrX-5_at,3,Control,2,12.00
AFFX-ThrX-5_at,3,Heat,1,16.60
AFFX-ThrX-5_at,3,Heat,2,18.80
AFFX-ThrX-5_at,3,Cold,1,13.80
AFFX-ThrX-5_at,3,Cold,2,17.70
AFFX-ThrX-5_at,4,Control,1,35.60
AFFX-ThrX-5_at,4,Control,2,17.00
AFFX-ThrX-5_at,4,Heat,1,12.50
AFFX-ThrX-5_at,4,Heat,2,18.70
AFFX-ThrX-5_at,4,Cold,1,20.40
AFFX-ThrX-5_at,4,Cold,2,9.60
AFFX-ThrX-5_at,5,Control,1,21.80
AFFX-ThrX-5_at,5,Control,2,15.00
AFFX-ThrX-5_at,5,Heat,1,8.00
AFFX-ThrX-5_at,5,Heat,2,18.30
AFFX-ThrX-5_at,5,Cold,1,13.90
AFFX-ThrX-5_at,5,Cold,2,15.40
AFFX-ThrX-5_at,6,Control,1,14.60
AFFX-ThrX-5_at,6,Control,2,13.60
AFFX-ThrX-5_at,6,Heat,1,17.50
AFFX-ThrX-5_at,6,Heat,2,17.50
AFFX-ThrX-5_at,6,Cold,1,21.30
AFFX-ThrX-5_at,6,Cold,2,17.70
AFFX-ThrX-M_at,1,Control,1,5.90
AFFX-ThrX-M_at,1,Control,2,56.70
AFFX-ThrX-M_at,1,Heat,1,62.30
AFFX-ThrX-M_at,1,Heat,2,59.00

AFFX-ThrX-M_at,1,Cold,1,54.10
AFFX-ThrX-M_at,1,Cold,2,53.20
AFFX-ThrX-M_at,2,Control,1,76.90
AFFX-ThrX-M_at,2,Control,2,38.70
AFFX-ThrX-M_at,2,Heat,1,17.20
AFFX-ThrX-M_at,2,Heat,2,78.60
AFFX-ThrX-M_at,2,Cold,1,62.90
AFFX-ThrX-M_at,2,Cold,2,43.40
AFFX-ThrX-M_at,3,Control,1,52.00
AFFX-ThrX-M_at,3,Control,2,60.20
AFFX-ThrX-M_at,3,Heat,1,74.90
AFFX-ThrX-M_at,3,Heat,2,47.20
AFFX-ThrX-M_at,3,Cold,1,13.00
AFFX-ThrX-M_at,3,Cold,2,20.60
AFFX-ThrX-M_at,4,Control,1,66.70
AFFX-ThrX-M_at,4,Control,2,6.10
AFFX-ThrX-M_at,4,Heat,1,11.20
AFFX-ThrX-M_at,4,Heat,2,18.10
AFFX-ThrX-M_at,4,Cold,1,53.20
AFFX-ThrX-M_at,4,Cold,2,29.70
AFFX-ThrX-M_at,5,Control,1,14.50
AFFX-ThrX-M_at,5,Control,2,42.80
AFFX-ThrX-M_at,5,Heat,1,44.90
AFFX-ThrX-M_at,5,Heat,2,60.70
AFFX-ThrX-M_at,5,Cold,1,40.30
AFFX-ThrX-M_at,5,Cold,2,34.80
AFFX-ThrX-M_at,6,Control,1,60.80
AFFX-ThrX-M_at,6,Control,2,17.80
AFFX-ThrX-M_at,6,Heat,1,68.80
AFFX-ThrX-M_at,6,Heat,2,38.20
AFFX-ThrX-M_at,6,Cold,1,55.30
AFFX-ThrX-M_at,6,Cold,2,53.10

TEST ONE-WAY ANOVA OUTPUT FILE

Name	p value for condition	Heat - Cold (p = 0.01)	Heat - Cold (p = 0.05)	Heat - Cold (p = 0.1)	Heat - Control (p = 0.01)	Heat - Control (p = 0.05)	Heat - Control (p = 0.1)	Cold - Control (p = 0.01)	Cold - Control (p = 0.05)	Cold - Control (p = 0.1)
AFFX-BioB-5_at	0.24353257	[-61.60949706, 154.94283039]	[-35.60528382, 128.93861716]	[-23.72404847, 117.0573818]	[-58.29838595, 158.2539415]	[-32.29417271, 132.24972827]	[-20.41293736, 120.36849291]	[-104.96505261, 111.58727484]	[-78.96083938, 85.5830616]	[-67.07960402, 73.70182624]
AFFX-BioB-M_at	0.59663903	[-151.4943285, 282.03877295]	[-99.43445297, 229.97889741]	[-75.64847535, 206.19291979]	[-178.19988406, 255.33321739]	[-126.14000852, 203.27334185]	[-102.35403091, 179.48736424]	[-243.47210628, 190.06099517]	[-191.41223074, 138.00111963]	[-167.62625313, 114.21514202]
AFFX-BioB-3_at	0.41437524	[-83.90605879, 195.26161435]	[-50.38281854, 161.73837409]	[-35.06616566, 146.42172122]	[-107.60050323, 171.5671699]	[-74.07726298, 138.04392965]	[-58.76061011, 122.72727677]	[-163.27828101, 115.88939212]	[-129.75504076, 82.36615187]	[-114.43838789, 67.049499]
AFFX-BioC-5_at	0.52575993	[-203.58465649, 388.7957676]	[-132.44994718, 317.66105829]	[-99.94874379, 285.1598549]	[-216.10687871, 376.27354538]	[-144.9721694, 305.13883607]	[-112.47096601, 272.63763267]	[-308.71243427, 283.66798982]	[-237.57772495, 212.53328051]	[-205.07652156, 180.03207712]
AFFX-BioC-3_at	0.41011583	[-204.63930978, 445.82819867]	[-126.52933982, 367.71822871]	[-90.84116389, 332.03005278]	[-220.58930978, 429.87819867]	[-142.47933982, 351.76822871]	[-106.79116389, 316.08005278]	[-341.18375423, 309.28375423]	[-263.07378427, 231.17378427]	[-227.38560834, 195.48560834]
AFFX-BioDn-5_at	0.28908168	[-164.77692358, 438.91025691]	[-92.28446712, 366.41780046]	[-59.16291374, 333.29624708]	[-191.81025691, 411.87692358]	[-119.31780046, 339.38446712]	[-86.19624708, 306.26291374]	[-328.87692358, 274.81025691]	[-256.38446712, 202.31780046]	[-223.26291374, 169.19624708]
AFFX-BioDn-3_at	0.27113168	[-1280.06153475, 3133.72820142]	[-750.04123517, 2603.70790184]	[-507.87678795, 2361.54345461]	[-1244.87820142, 3168.91153475]	[-714.85790184, 2638.89123517]	[-472.69345461, 2396.72678795]	[-2171.71153475, 2242.07820142]	[-1641.69123517, 1712.05790184]	[-1399.52678795, 1469.89345461]
AFFX-CreX-5_at	0.43169993	[-2180.34879816, 4149.35990927]	[-1420.25955379, 3389.2706649]	[-1072.97739904, 3041.98851015]	[-2026.08768705, 4303.62102039]	[-1265.99844268, 3543.53177601]	[-918.71628792, 3196.24962126]	[-3010.59324261, 3319.11546483]	[-2250.50399824, 2559.02622046]	[-1903.22184348, 2211.7440657]
AFFX-CreX-3_at	0.42987069	[-2749.89818953, 5577.72041175]	[-1749.89429952, 4577.71652174]	[-1292.99597046, 4120.81819269]	[-2754.72596731, 5572.89263397]	[-1754.7220773, 4572.88874397]	[-1297.82374824, 4115.99041491]	[-4168.63707842, 4158.98152286]	[-3168.63318841, 3158.97763285]	[-2711.73485935, 2702.0793038]
AFFX-DapX-5_at	0.71919154	[-21.26180897, 28.50625342]	[-15.28551991, 22.52996435]	[-12.55497404, 19.79941849]	[-18.95069786, 30.81736453]	[-12.9744088, 24.84107546]	[-10.24386293, 22.1105296]	[-22.57292008, 27.19514231]	[-16.59663102, 21.21885324]	[-13.86608515, 18.48830737]
AFFX-DapX-M_at	0.08185544	[-9.65480618, 53.5770284]	[-2.06174944, 45.98397166]	[1.407492, 42.51473022]	[-25.1270284, 38.10480618]	[-17.53397166, 30.51174944]	[-14.06473022, 27.042508]	[-47.08813951, 16.14369507]	[-39.49508277, 8.55063833]	[-36.02584133, 5.08139688]
AFFX-DapX-3_at	0.10017086	[-4.4658039, 5.52135946]	[-3.2665172, 4.32207276]	[-2.71856725, 3.7741228]	[-1.82691501, 8.16024835]	[-0.62762832, 6.96096165]	[-0.07967836, 6.41301169]	[-2.35469279, 7.63247057]	[-1.15540609, 6.43318387]	[-0.60745614, 5.88523391]
AFFX-LysX-5_at	0.86836897	[-12.3651477, 14.87625881]	[-9.0939229, 11.60503402]	[-7.59931157, 10.11042268]	[-14.48181437, 12.75959215]	[-11.21058957, 9.48836735]	[-9.71597824, 7.99375602]	[-15.73736993, 11.50403659]	[-12.46614513, 8.23281179]	[-10.9715338, 6.73820046]
AFFX-LysX-M_at	0.16022286	[-9.75716852, 34.16827963]	[-4.48247702, 28.89358814]	[-2.07248867, 26.48359978]	[-11.74605741, 32.17939074]	[-6.47136591, 26.90469925]	[-4.06137756, 24.49471089]	[-23.95161297, 19.97383519]	[-18.67692147, 14.69914369]	[-16.26693311, 12.28915534]
AFFX-LysX-3_at	0.55522731	[-18.15072762, 17.62850539]	[-13.85425657, 13.33203435]	[-11.89121377, 11.36899154]	[-13.00628317, 22.77294984]	[-8.70981213, 18.47647879]	[-6.74676932, 16.51343599]	[-12.74517206, 23.03406095]	[-8.44870102, 18.7375899]	[-6.48565821, 16.7745471]
AFFX-PheX-5_at	0.75977661	[-2.74953001, 1.81619668]	[-2.20126469, 1.26793136]	[-1.95076416, 1.01743083]	[-2.36064112, 2.20508557]	[-1.81237581, 1.65682025]	[-1.56187527, 1.40631972]	[-1.89397446, 2.67175223]	[-1.34570914, 2.12348692]	[-1.09520861, 1.87298638]
AFFX-PheX-M_at	0.12279424	[-1.53957245, 6.45068356]	[-0.58008001, 5.49119112]	[-0.14169122, 5.05280233]	[-2.10068356, 5.88957245]	[-1.14119112, 4.93008001]	[-0.70280233, 4.49169122]	[-4.55623911, 3.43401689]	[-3.59674667, 2.47452445]	[-3.15835789, 2.03613566]
AFFX-PheX-3_at	0.0983472	[-17.72577462, 54.4035524]	[-9.06428193, 45.74205971]	[-5.10687579, 41.78465357]	[-12.67021906, 59.45910795]	[-4.00872638, 50.79761527]	[-0.05132024, 46.84020913]	[-31.00910795, 41.12021906]	[-22.34761527, 32.45872638]	[-18.39020913, 28.50132024]
AFFX-ThrX-5_at	0.16039673	[-6.76397456, 13.81953012]	[-4.29224936, 11.34780492]	[-3.16292664, 10.2184822]	[-4.18064123, 16.40286345]	[-1.70891603, 13.93113825]	[-0.57959331, 12.80181553]	[-7.70841901, 12.87508568]	[-5.23669381, 10.40336048]	[-4.10737109, 9.27403775]
AFFX-ThrX-M_at	0.76935162	[-17.9959178, 25.38480669]	[-12.78663825, 20.17552714]	[-10.40653639, 17.79542528]	[-17.41258446, 25.96814002]	[-12.20330492, 20.75886048]	[-9.82320306, 18.37875861]	[-21.10702891, 22.27369558]	[-15.89774936, 17.06441603]	[-13.5176475, 14.68431417]

TEST TWO-WAY ANOVA OUTPUT FILE

Probe	p value for condition	p value for time	p value for interaction
AFFX-BioB-5_at	0.05794984	0.13501568	0.24353257
AFFX-BioB-M_at	0.35676659	0.30461054	0.59663903
AFFX-BioB-3_at	0.06939791	0.21827141	0.41437524
AFFX-BioC-5_at	0.24299456	0.30101028	0.52575993
AFFX-BioC-3_at	0.22649861	0.31565169	0.41011583
AFFX-BioDn-5_at	0.11384801	0.19242631	0.28908168
AFFX-BioDn-3_at	0.16022979	0.16946863	0.27113168
AFFX-CreX-5_at	0.13472166	0.09849027	0.43169993
AFFX-CreX-3_at	0.13544086	0.08311014	0.42987069
AFFX-DapX-5_at	0.95629408	0.82901326	0.71919154
AFFX-DapX-M_at	0.94023323	0.68961177	0.08185544
AFFX-DapX-3_at	0.30338196	0.67242893	0.10017086
AFFX-LysX-5_at	0.84402251	0.77746585	0.86836897
AFFX-LysX-M_at	0.79542281	0.17446481	0.16022286
AFFX-LysX-3_at	0.70302937	0.56640906	0.55522731
AFFX-PheX-5_at	0.63177669	0.80000401	0.75977661
AFFX-PheX-M_at	0.57126931	0.8818067	0.12279424
AFFX-PheX-3_at	0.53834261	0.46861711	0.0983472
AFFX-ThrX-5_at	0.52682431	0.14079342	0.16039673
AFFX-ThrX-M_at	0.70419305	0.59522396	0.76935162
No. of Genes (P <= 0.01)	0	0	0